



AutoHotkey

Automation. Hotkeys. Scripting.

Version **AHK_H 2.0-alpha**

<https://autohotkey.com>

© 2017 Hotkeyit, Steve Gray, Chris Mallett, portions © [AutoIt Team](#) and various others

Software License: [GNU General Public License](#)

Quick Reference

Fundamentals:

- [Tutorial for beginners](#)
- [Text editors with AutoHotkey support](#)
- [Frequently asked questions](#)
- [Scripts](#)
 - [Commands](#)
 - [Variables and expressions](#)
 - [Functions](#)
 - [Objects](#)
 - [Interactive debugging](#)

Keyboard and mouse:

- [Hotkeys \(mouse, joystick and keyboard shortcuts\)](#)
- [Hotstrings and auto-replace](#)
- [Remapping keys and buttons](#)
- [List of keys, mouse buttons and joystick controls](#)

Other:

- [DllCall](#)
- [RegEx quick reference](#)

Acknowledgements

A special thanks to Jonathan Bennett, whose generosity in releasing AutoIt v2 as free software in 1999 served as an inspiration and time-saver for myself and many others worldwide. In addition, many of AutoHotkey's enhancements to the AutoIt v2 command set, as well as the Window Spy and the old script compiler, were adapted directly from the AutoIt v3 source code. So thanks to Jon and the other AutoIt authors for those as well.

Finally, AutoHotkey would not be what it is today without [these other individuals](#).

~ Chris Mallett

Hotkeys (Mouse, Joystick and Keyboard Shortcuts)

Table of Contents

- [Introduction and Simple Examples](#)
- [Table of Hotkey Prefix Symbols \(Modifiers\)](#)
- [Context-sensitive Hotkeys](#)
- [Custom Combinations](#)
- [Other Features](#)
- [Mouse Wheel Hotkeys](#)
- [Hotkey Tips and Remarks](#)
- [Alt-Tab Hotkeys](#)
- [Function Hotkeys](#)

Introduction and Simple Examples

Hotkeys are sometimes referred to as shortcut keys because of their ability to easily trigger an action (such as launching a program or [keyboard macro](#)). In the following example, the hotkey Win+N is configured to launch Notepad. The pound sign [#] stands for the Windows key, which is known as a *modifier*:

```
#n::  
Run Notepad  
return
```

In the final line above, `return` serves to finish the hotkey. However, if a hotkey needs to execute only a single line, that line can be listed to the right of the double-colon. In other words, the `return` is implicit:

```
#n::Run Notepad
```

To use more than one modifier with a hotkey, list them consecutively (the order does not matter). The following example uses `^!s` to indicate Control+Alt+S:

```
^!s::  
Send Sincerely,{enter}John Smith ; This line  
sends keystrokes to the active (foremost)  
window.  
return
```

You can use the following modifier symbols to define hotkeys:

Symbol	Description
#	Win (Windows logo key). For Windows Vista and later, hotkeys that include the Windows key (e.g. #a) will wait for the Windows key to be released before sending any text containing an "L" keystroke. This prevents usages of <code>Send</code> within such a hotkey from locking the PC. This behavior applies to all sending modes except <code>SendPlay</code> (which doesn't need it) and <code>blind mode</code> .
!	Alt
^	Control
+	Shift
&	An ampersand may be used between any two keys or mouse buttons to combine them into a custom hotkey. See below for details.
<	Use the left key of the pair. e.g. <!a is the same as !a except that only the left Alt key will trigger it.
>	Use the right key of the pair.
<^>!	<p>AltGr (alternate graving). If your keyboard layout has an AltGr key instead of a right-Alt key, this series of symbols can usually be used to stand for AltGr. For example:</p> <pre><^>!m::MsgBox You pressed AltGr+m. <^<!m::MsgBox You pressed LeftControl+LeftAlt+m.</pre> <p>Alternatively, to make AltGr itself into a hotkey, use the following hotkey (without any hotkeys like the above present):</p> <pre>LControl & RAlt::MsgBox You pressed AltGr itself.</pre>

Wildcard: Fire the hotkey even if extra modifiers are being held down. This is often used in conjunction with [remapping](#) keys or buttons. For example:

*

```
*#c::Run Calc.exe ; Win+C,  
Shift+Win+C, Ctrl+Win+C, etc. will all  
trigger this hotkey.  
*ScrollLock::Run Notepad ; Pressing  
ScrollLock will trigger this hotkey  
even when modifier key(s) are down.
```

When the hotkey fires, its key's native function will not be blocked (hidden from the system). In both of the below examples, the user's click of the mouse button will be sent to the active window:

```
~RButton::MsgBox You clicked the right  
mouse button.  
~RButton & C::MsgBox You pressed C  
while holding down the right mouse  
button.
```

Unlike the other prefix symbols, the tilde prefix is allowed to be present on some of a hotkey's [variants](#) but absent on others. However, if a tilde is applied to the [prefix key](#) of any custom combination which has not been turned off or suspended, it affects the behavior of that prefix key for *all* combinations.

Special hotkeys that are substitutes for [alt-tab](#) always ignore the tilde prefix.

~

If the tilde prefix is applied to a custom modifier key ([prefix key](#)) which is also used as its own hotkey, that hotkey will fire when the key is pressed instead of being delayed until the key is released. For example, the `~RButton` hotkey above is fired as soon as the button is pressed. Prior to v1.1.14 (or without the tilde prefix), it

was fired when the button was released, but only if the *RButton & C* combination was not activated.

If the tilde prefix is applied only to the custom combination and not the non-combination hotkey, the key's native function will still be blocked. For example, in the script below, holding `AppsKey` will show the `ToolTip` and will not trigger a context menu:

```
AppsKey::ToolTip Press < or > to cycle
through windows.
AppsKey Up::ToolTip
~AppsKey & <::Send !+{Esc}
~AppsKey & >::Send !{Esc}
```

This is usually only necessary if the script uses the `Send` command to send the keys that comprise the hotkey itself, which might otherwise cause it to trigger itself. The `$` prefix forces the `keyboard hook` to be used to implement this hotkey, which as a side-effect prevents the `Send` command from triggering it. The `$` prefix is equivalent to having specified `#UseHook` somewhere above the definition of this hotkey.

\$

The `$` prefix has no effect for mouse hotkeys, since they always use the mouse hook. It also has no effect for hotkeys which already require the keyboard hook, including any keyboard hotkeys with the `tilde (~)` or `wildcard (*)` modifiers, key-up hotkeys and custom combinations.

`#InputLevel` and `SendLevel` provide additional control over which hotkeys and hotstrings are triggered by the `Send` command.

The word `UP` may follow the name of a hotkey to cause the hotkey to fire upon release of the key rather than when the key is pressed down. The following example `remaps` `LWin` to become `LControl`:

UP

```
*LWin::Send {LControl Down}
*LWin Up::Send {LControl Up}
```

"Up" can also be used with normal hotkeys as in this example: `^!r Up::MsgBox You pressed and released Ctrl+Alt+R.` It also works with [combination hotkeys](#) (e.g. `F1 & e Up::`)

Limitations: 1) "Up" does not work with [joystick buttons](#); 2) An "Up" hotkey without a normal/down counterpart hotkey will completely take over that key to prevent it from getting stuck down. One way to prevent this is to add a [tilde prefix](#) (e.g. `~LControl up::`)

On a related note, a technique similar to the above is to make a hotkey into a prefix key. The advantage is that although the hotkey will fire upon release, it will do so only if you did not press any other key while it was held down. For example:

```
LControl & F1::return ; Make left-
control a prefix by using it in front
of "&" at least once.
LControl::MsgBox You released LControl
without having used it to modify any
other key.
```

(See the [Key List](#) for a complete list of keyboard keys and mouse/joystick buttons)

Multiple hotkeys can be stacked vertically to have them perform the same action. For example:

```
^Numpad0::
^Numpad1::
```

```
MsgBox Pressing either Control+Numpad0 or  
Control+Numpad1 will display this message.  
return
```

A key or key-combination can be disabled for the entire system by having it do nothing. The following example disables the right-side Windows key:

```
Rwin::return
```

Context-sensitive Hotkeys

The directives `#IfWinActive/Exist` and `#If` can be used to make a hotkey perform a different action (or none at all) depending on a specific condition. For example:

```
#IfWinActive, ahk_class Notepad
^a::MsgBox You pressed Ctrl-A while Notepad is
active. Pressing Ctrl-A in any other window
will pass the Ctrl-A keystroke to that window.
#c::MsgBox You pressed Win-C while Notepad is
active.

#IfWinActive
#c::MsgBox You pressed Win-C while any window
except Notepad is active.

#If MouseIsOver("ahk_class Shell_TrayWnd")
WheelUp::Send {Volume_Up} ; Wheel over
taskbar: increase/decrease volume.
WheelDown::Send {Volume_Down} ;
```

Custom Combinations

You can define a custom combination of two keys (except joystick buttons) by using "&" between them. In the below example, you would hold down Numpad0 then press the second key to trigger the hotkey:

```
Numpad0 & Numpad1::MsgBox You pressed Numpad1  
while holding down Numpad0.  
Numpad0 & Numpad2::Run Notepad
```

The prefix key loses its native function: In the above example, Numpad0 becomes a *prefix key*; but this also causes Numpad0 to lose its original/native function when it is pressed by itself. To avoid this, a script may configure Numpad0 to perform a new action such as one of the following:

```
Numpad0::WinMaximize A ; Maximize the  
active/foreground window.  
Numpad0::Send {Numpad0} ; Make the release of  
Numpad0 produce a Numpad0 keystroke. See  
comment below.
```

Fire on release: The presence of one of the above custom combination hotkeys causes the *release* of Numpad0 to perform the indicated action, but only if you did not press any other keys while Numpad0 was being held down. This behaviour can be avoided by applying the *tilde prefix* to either hotkey.

Modifiers: Unlike a normal hotkey, custom combinations act as though they have the *wildcard (*)* modifier by default. For example, `1 & 2::` will activate

even if Ctrl or Alt is held down when 1 and 2 are pressed, whereas `^1::` would be activated only by Ctrl+1 and not Ctrl+Alt+1.

For standard modifier keys, normal hotkeys typically work as well or better than "custom" combinations. For example, `<+s::` is recommended over `LShift & S::`.

Combinations of three or more keys are not supported. Combinations which your keyboard hardware supports can usually be detected by using `#if` and `GetKeyState`, but the results may be inconsistent. For example:

```
; Press AppsKey and Alt in any order, then slash (/).  
#if GetKeyState("AppsKey", "P")  
Alt & /::MsgBox Hotkey activated.  
  
; If the keys are swapped, Alt must be pressed first (use one at a time):  
#if GetKeyState("Alt", "P")  
AppsKey & /::MsgBox Hotkey activated.  
  
; [ & ] & \::  
#if GetKeyState("[") && GetKeyState("]")  
\::MsgBox
```

Other Features

Numlock, Capslock, and Scrolllock: These keys may be forced to be "AlwaysOn" or "AlwaysOff". For example: `SetNumlockState AlwaysOn`.

Overriding Explorer's hotkeys: Windows' built-in hotkeys such as Win-E (#e) and Win-R (#r) can be individually overridden simply by assigning them to an action in the script. See the [override](#) page for details.

Substitutes for Alt-Tab: Hotkeys can provide an alternate means of alt-tabbing. For example, the following two hotkeys allow you to alt-tab with your right hand:

```
RControl & RShift::AltTab ; Hold down right-control then press right-shift repeatedly to move forward.  
RControl & Enter::ShiftAltTab ; Without even having to release right-control, press Enter to reverse direction.
```

For more details, see [Alt-Tab](#).

Mouse Wheel Hotkeys

Hotkeys that fire upon turning the mouse wheel are supported via the key names `WheelDown` and `WheelUp`. `WheelLeft` and `WheelRight` are also supported, but have no effect on operating systems older than Windows Vista. Here are some examples of mouse wheel hotkeys:

```
MButton & WheelDown::MsgBox You turned the  
mouse wheel down while holding down the middle  
button.  
^!WheelUp::MsgBox You rotated the wheel up  
while holding down Control+Alt.
```

The built-in variable `A_EventInfo` contains the amount by which the wheel was turned, which is typically 120. However, `A_EventInfo` can be greater or less than 120 under the following circumstances:

- If the mouse hardware reports distances of less than one notch, `A_EventInfo` may be less than 120;
- If the wheel is being turned quickly (depending on the type of mouse), `A_EventInfo` may be greater than 120. A hotkey like the following can help analyze your mouse: `-wheelDown::ToolTip %A_EventInfo%`

Some of the most useful hotkeys for the mouse wheel involve alternate modes of scrolling a window's text. For example, the following pair of hotkeys scrolls horizontally instead of vertically when you turn the wheel while holding down the left Control key:

```

~LControl & WheelUp:: ; Scroll left.
ControlGetFocus, fcontrol, A
Loop 2 ; <-- Increase this value to scroll
faster.
    SendMessage, 0x114, 0, 0, %fcontrol%, A ;
0x114 is WM_HSCROLL and the 0 after it is
SB_LINELEFT.
return

~LControl & WheelDown:: ; Scroll right.
ControlGetFocus, fcontrol, A
Loop 2 ; <-- Increase this value to scroll
faster.
    SendMessage, 0x114, 1, 0, %fcontrol%, A ;
0x114 is WM_HSCROLL and the 1 after it is
SB_LINERIGHT.
return

```

Finally, since mouse wheel hotkeys generate only down-events (never up-events), they cannot be used as [key-up hotkeys](#).

Hotkey Tips and Remarks

Each numpad key can be made to launch two different hotkey subroutines depending on the state of Numlock. Alternatively, a numpad key can be made to launch the same subroutine regardless of the Numlock state. For example:

```
NumpadEnd::  
Numpad1::  
MsgBox, This hotkey is launched regardless of  
whether Numlock is on.  
return
```

If the [tilde \(~\) operator](#) is used with a [prefix key](#) even once, it changes the behavior of that prefix key for all combinations. For example, in both of the below hotkeys, the active window will receive all right-clicks even though only one of the definitions contains a tilde:

```
~RButton & LButton::MsgBox You pressed the left  
mouse button while holding down the right.  
RButton & WheelUp::MsgBox You turned the mouse  
wheel up while holding down the right button.
```

The [Suspend](#) command can temporarily disable all hotkeys except for ones you make exempt. For greater selectivity, use [#IfWinActive/Exist](#).

By means of the [Hotkey](#) command, hotkeys can be created dynamically while the script is running. The Hotkey command can also modify, disable, or enable the script's existing hotkeys individually.

Joystick hotkeys do not currently support modifier prefixes such as ^ (Control) and # (Win). However, you can use [GetKeyState](#) to mimic this effect as shown in the following example:

```
Joy2::
if not GetKeyState("Control") ; Neither the
left nor right Control key is down.
return ; i.e. Do nothing.
MsgBox You pressed the first joystick's second
button while holding down the Control key.
return
```

There may be times when a hotkey should wait for its own modifier keys to be released before continuing. Consider the following example:

```
^!s::Send {Delete}
```

Pressing Control-Alt-S would cause the system to behave as though you pressed Control-Alt-Delete (due to the system's aggressive detection of Ctrl-Alt-Delete). To work around this, use [KeyWait](#) to wait for the keys to be released; for example:

```
^!s::
KeyWait Control
KeyWait Alt
Send {Delete}
return
```

If a hotkey label like `#z::` produces an error like "Invalid Hotkey", your

system's keyboard layout/language might not have the specified character ("Z" in this case). Try using a different character that you know exists in your keyboard layout.

A hotkey label can be used as the target of a [Gosub](#) or [Goto](#). For example:

```
Gosub ^!s.
```

One common use for hotkeys is to start and stop a repeating action, such as a series of keystrokes or mouse clicks. For an example of this, see [this FAQ topic](#).

Finally, each script is [quasi multi-threaded](#), which allows a new hotkey to be launched even when a previous hotkey subroutine is still running. For example, new hotkeys can be launched even while a [MsgBox](#) is being displayed by the current hotkey.

Alt-Tab Hotkeys

Each Alt-Tab hotkey must be a combination of two keys, which is typically achieved via the ampersand symbol (&). In the following example, you would hold down the right Alt key and press J or K to navigate the alt-tab menu:

```
RAlt & j::AltTab  
RAlt & k::ShiftAltTab
```

AltTab and *ShiftAltTab* are two of the special commands that are only recognized when used on the same line as a hotkey. Here is the complete list:

AltTab: If the alt-tab menu is visible, move forward in it. Otherwise, display the menu (only if the hotkey is an "&" combination of two keys; otherwise, it does nothing).

ShiftAltTab: Same as above except move backward in the menu.

AltTabAndMenu: If the alt-tab menu is visible, move forward in it. Otherwise, display the menu.

AltTabMenuDismiss: Close the Alt-tab menu.

To illustrate the above, the mouse wheel can be made into an entire substitute for Alt-tab. With the following hotkeys in effect, clicking the middle button displays the menu and turning the wheel navigates through it:

```
MButton::AltTabMenu
```

```
WheelDown::AltTab
WheelUp::ShiftAltTab
```

To cancel a hotkey-invoked Alt-tab menu without activating the selected window, use a hotkey such as the following. It might require adjustment depending on: 1) the means by which the alt-tab menu was originally displayed; and 2) whether the script has the [keyboard hook](#) installed.

```
LCtrl & CapsLock::AltTab
!MButton:: ; Middle mouse button. The ! prefix
makes it fire while the Alt key is down (which
it is if the alt-tab menu is visible).
if WinExist("ahk_class #32771") ; Indicates
that the alt-tab menu is present on the screen.
    Send !{Escape}{Alt up}
return
```

Currently, all special Alt-tab actions must be assigned directly to a hotkey as in the examples above (i.e. they cannot be used as though they were commands). They are **not affected** by `#IfWin` or `#If`.

Custom alt-tab actions can also be created via hotkeys. In the following example, you would press F1 to display the menu and advance forward in it. Then you would press F2 to activate the selected window (or press Escape to cancel):

```
*F1::Send {Alt down}{tab} ; Asterisk is
required in this case.
!F2::Send {Alt up} ; Release the Alt key,
which activates the selected window.
~*Escape::
if WinExist("ahk_class #32771")
```

```
Send {Escape}{Alt up} ; Cancel the menu  
without activating the selected window.  
return
```

Function Hotkeys

One or more hotkeys can be assigned a [function](#) by simply defining it immediately after the hotkey label as in this example:

```
; Ctrl+Shift+O to open containing folder in Explorer.
; Ctrl+Shift+E to open folder with current file selected.
; Supports SciTE and Notepad++.
^+o::
^+e::
    editor_open_folder() {
        WinSetTitle, path, A
        if RegExMatch(path, "\*?\K(.*)\\\[^\\"+
(=? [-*] )", path)
            if (FileExist(path) && A_ThisHotkey
= "^+e")
                Run explorer.exe
                /select`, "%path%"
            else
                Run explorer.exe "%path1%"
    }
```

There must only be whitespace, comments or directives between the hotkey labels or label and the function. No label is created for hotkeys defined this way; however, the [auto-execute section](#) ends at the first hotkey even if it is assigned a function.

The main benefit of using a function is that local variables can be used, which avoids conflicts when two or more hotkeys use the same variable names for

different purposes. It also encourages self-documenting hotkeys, like in the code above where the function name describes the hotkey.

The `Hotkey` command can also be used to assign a function or function object to a hotkey.

Hotstrings and Auto-replace

Introduction and Simple Examples

Although hotstrings are mainly used to expand abbreviations as you type them (auto-replace), they can also be used to launch any scripted action. In this respect, they are similar to [hotkeys](#) except that they are typically composed of more than one character (that is, a string).

To define a hotstring, enclose the triggering abbreviation between pairs of colons as in this example:

```
::btw::by the way
```

In the above example, the abbreviation btw will be automatically replaced with "by the way" whenever you type it (however, by default you must type an [ending character](#) after typing btw, such as a space, period, or enter).

The "by the way" example above is known as an auto-replace hotstring because the typed text is automatically erased and replaced by the string specified after the second pair of colons. By contrast, a hotstring may also be defined to perform any custom action as in the following examples. Note that the commands must appear beneath the hotstring:

```
::btw::  
MsgBox You typed "btw".  
return
```

```
 :*:]d:: ; This hotstring replaces "]d" with  
 the current date and time via the commands  
 below.  
 FormatTime, CurrentDateTime,, M/d/yyyy h:mm tt  
 ; It will look like 9/1/2005 3:53 PM  
 SendInput %CurrentDateTime%  
 return
```

Even though the two examples above are not auto-replace hotstrings, the abbreviation you type is erased by default. This is done via automatic backspacing, which can be disabled via the [b0 option](#).

Ending Characters

Unless the [asterisk option](#) is in effect, you must type an *ending character* after a hotstring's abbreviation to trigger it. Ending characters initially consist of the following: `-()[]{}:;'"^,.?!\`n `t` (note that ``n` is Enter, ``t` is Tab, and there is a plain space between ``n` and ``t`). This set of characters can be changed by editing the following example, which sets the new ending characters for all hotstrings, not just the ones beneath it:

```
#Hotstring EndChars -()[]{}:;'"^,.?!\`n `t
```

Options

A hotstring's default behavior can be changed in two possible ways:

1. The `#Hotstring` directive, which affects all hotstrings physically beneath that point in the script. The following example puts the C and R options into effect: `#Hotstring c r|`.
2. Putting options inside a hotstring's first pair of colons. The following example puts the C and * options into effect for a single hotstring:

```
:c* :j@::john@somedomain.com ; Case sensitive and  
"ending character not required".|
```

The list below describes each option. When specifying more than one option using the methods above, spaces optionally may be included between them.

* (asterisk): An ending character (e.g. space, period, or enter) is not required to trigger the hotstring. For example:

```
:*:j@::jsmith@somedomain.com
```

The example above would send its replacement the moment you type the @ character. When using the `#Hotstring` directive, use `*0` to turn this option back off.

? (question mark): The hotstring will be triggered even when it is inside another word; that is, when the character typed immediately before it is alphanumeric. For example, if `:?:al::airline` is a hotstring, typing "practical " would

produce "practicairline ". Use **?0** to turn this option back off.

B0 (B followed by a zero): Automatic backspacing is not done to erase the abbreviation you type. Use a plain **B** to turn backspacing back on after it was previously turned off. A script may also do its own backspacing via `{bs 5}`, which sends 5 backspaces. Similarly, it may send left-arrow keystrokes via `{left 5}`. For example, the following hotstring produces "``" and moves the caret 5 places to the left (so that it's between the tags):

```
:*b0:<em>::</em>{left 5}
```

C: Case sensitive: When you type an abbreviation, it must exactly match the case defined in the script. Use **C0** to turn case sensitivity back off.

C1: Do not conform to typed case. Use this option to make [auto-replace hotstrings](#) case insensitive and prevent them from conforming to the case of the characters you actually type. Case-conforming hotstrings (which are the default) produce their replacement text in all caps if you type the abbreviation in all caps. If you type only the first letter in caps, the first letter of the replacement will also be capitalized (if it is a letter). If you type the case in any other way, the replacement is sent exactly as defined. When using the [#Hotstring directive](#), **C0** can be used to turn this option back off, which makes hotstrings conform again.

Kn: Key-delay: This rarely-used option sets the delay between keystrokes produced by auto-backspacing or [auto-replacement](#). Specify the new delay for **n**; for example, specify `k10` to have a 10ms delay and `k-1` to have no delay. The exact behavior of this option depends on which [sending mode](#) is in effect:

- SI (SendInput): Key-delay is ignored because a delay is not possible in this mode. The exception to this is when SendInput is `unavailable`, in which case hotstrings revert to SendPlay mode below (which does obey key-delay).
- SP (SendPlay): A delay of length zero is the default, which for SendPlay is the same as -1 (no delay). In this mode, the delay is actually a `PressDuration` rather than a delay between keystrokes.
- SE (SendEvent): A delay of length zero is the default. Zero is recommended for most purposes since it is fast but still cooperates well with other processes (due to internally doing a `Sleep 0`). Specify k-1 to have no delay at all, which is useful to make auto-replacements faster if your CPU is frequently under heavy load. When set to -1, a script's process-priority becomes an important factor in how fast it can send keystrokes. To raise a script's priority, use `ProcessSetPriority High`.

O: Omit the ending character of `auto-replace hotstrings` when the replacement is produced. This is useful when you want a hotstring to be kept unambiguous by still requiring an ending character, but don't actually want the ending character to be shown on the screen. For example, if `:o:ar::aristocrat` is a hotstring, typing "ar" followed by the spacebar will produce "aristocrat" with no trailing space, which allows you to make the word plural or possessive without having to backspace. Use `O0` (the letter O followed by a zero) to turn this option back off.

Pn: The `priority` of the hotstring (e.g. P1). This rarely-used option has no effect on `auto-replace hotstrings`.

R: Send the replacement text raw; that is, exactly as it appears rather than translating {Enter} to an ENTER keystroke, ^c to Control-C, etc. This option is put into effect automatically for hotstrings that have a [continuation section](#). Use **R0** to turn this option back off.

SI or **SP** or **SE:** Sets the method by which [auto-replace hotstrings](#) send their keystrokes. These options are mutually exclusive: only one can be in effect at a time. The following describes each option:

- **SI** stands for [SendInput](#), which typically has superior speed and reliability than the other modes. Another benefit is that like [SendPlay](#) below, [SendInput](#) postpones anything you type during a hotstring's [auto-replacement text](#). This prevents your keystrokes from being interspersed with those of the replacement. When [SendInput](#) is [unavailable](#), hotstrings automatically use [SendPlay](#) instead.
- **SP** stands for [SendPlay](#), which may allow hotstrings to work in a broader variety of games.
- **SE** stands for [SendEvent](#).

If none of the above options are used, the default mode is [SendInput](#). However, unlike the **SI** option, [SendEvent](#) is used instead of [SendPlay](#) when [SendInput](#) is unavailable.

Z: This rarely-used option resets the hotstring recognizer after each triggering of the hotstring. In other words, the script will begin waiting for an entirely new hotstring, eliminating from consideration anything you previously typed. This can prevent unwanted triggerings of hotstrings. To illustrate, consider the

following hotstring:

```
:b0*?:11::  
SendInput xx  
return
```

Since the above lacks the Z option, typing 111 (three consecutive 1's) would trigger the hotstring twice because the middle 1 is the *last* character of the first triggering but also the *first* character of the second triggering. By adding the letter Z in front of b0, you would have to type four 1's instead of three to trigger the hotstring twice. Use **Z0** to turn this option back off.

Long Replacements

Hotstrings that produce a large amount of replacement text can be made more readable and maintainable by using a [continuation section](#). For example:

```
::text1::  
(  
Any text between the top and bottom parentheses  
is treated literally.  
By default, the hard carriage return (Enter)  
between the previous line and this one is also  
preserved.  
    By default, the indentation (tab) to the  
left of this line is preserved.  
  
See continuation section for how to change  
these default behaviors.  
)
```

The presence of a continuation section also causes the hotstring to default to [raw mode](#). The only way to override this special default is to specify the [r0 option](#) in each hotstring that has a continuation section (e.g. `:r0:text1::`).

Context-sensitive Hotstrings

The directives `#IfWinActive/Exist` can be used to make selected hotstrings context sensitive. Such hotstrings send a different replacement, perform a different action, or do nothing at all depending on the type of window that is active or exists. For example:

```
#IfWinActive ahk_class Notepad
::btw::This replacement text will appear only
in Notepad.
#IfWinActive
::btw::This replacement text appears in windows
other than Notepad.
```

AutoCorrect

The following script uses hotstrings to correct about 4700 common English misspellings on-the-fly. It also includes a Win+H hotkey to make it easy to add more misspellings:

Download: [AutoCorrect.abk](#) (127 KB)

Author: [Jim Bianco](#) and [Wikipedia's Lists of Common Misspellings](#)

Remarks

Variable references such as `%MyVar%` are not currently supported within the replacement text. To work around this, don't make such hotstrings [auto-replace](#). Instead, use the [SendInput](#) command beneath the abbreviation, followed by a line containing only the word `Return`.

To send an extra space or tab after a replacement, include the space or tab at the end of the replacement but make the last character an accent/backtick (```). For example:

```
:*:btw::By the way `
```

Any click of the left or right mouse button will reset the hotstring recognizer. In other words, the script will begin waiting for an entirely new hotstring, eliminating from consideration anything you previously typed (if this is undesirable, specify the line `#Hotstring NoMouse` anywhere in the script). This "reset upon mouse click" behavior is the default because each click typically moves the text insertion point (caret) or sets keyboard focus to a new control/field. In such cases, it is usually desirable to: 1) fire a hotstring even if it lacks the [question mark option](#); 2) prevent a firing when something you type after clicking the mouse accidentally forms a valid abbreviation with what you typed before.

The built-in variable `A_EndChar` contains the ending character that you typed to trigger the most recent non-auto-replace hotstring. If no ending character was

required (due to the `* option`), it will be blank. `A_EndChar` is useful when making hotstrings that use the `Send` command or whose behavior should vary depending on which ending character you typed. To send the ending character itself, use `SendRaw %A_EndChar%` (`SendRaw` is used because characters such as `!{}` would not be sent correctly by the normal `Send` command).

Although single-colons within hotstring definitions do not need to be `escaped`, backticks and those semicolons having a space or tab to their left require it. See `escape sequences` for a complete list.

Although the `Send command's` special characters such as `{Enter}` are supported in `auto-replacement text` (unless the `raw option` is used), the hotstring abbreviations themselves do not use this. Instead, specify ``n` for the ENTER key and ``t` (or a literal tab) for TAB (see `escape sequences` for a complete list). For example, the hotstring `:*:ab t:::` would be triggered when you type "ab" followed by a tab.

Spaces and tabs are treated literally within hotstring definitions. For example, the following would produce two different results: `::btw::by the way` and `::btw:: by the way`.

Each hotstring abbreviation can be no more than 40 characters long. The program will warn you if this length is exceeded. By contrast, the length of hotstring's replacement text is limited to about 5000 characters when the `sending mode` is at its default of `SendInput`. That limit can be increased to 16,383 characters by switching to one of the other `sending modes`. Furthermore, an unlimited amount of text can be sent by using `SendPlay %MyVariable%` in

the body of the hotstring.

The order in which hotstrings are defined determines their precedence with respect to each other. In other words, if more than one hotstring matches something you type, only the one listed first in the script will take effect. Related topic: [context-sensitive hotstrings](#).

Any backspacing you do is taken into account for the purpose of detecting hotstrings. However, the use of arrow keys, PageUp, PageDown, Home, and End to navigate within an editor will cause the hotstring recognition process to reset. In other words, it will begin waiting for an entirely new hotstring.

A hotstring may be typed even when the active window is ignoring your keystrokes. In other words, the hotstring will still fire even though the triggering abbreviation is never visible. In addition, you may still press the backspace key to undo the most recently typed keystroke (even though you can't see the effect).

It is possible to [Gosub](#) or [Goto](#) a hotstring label by including its first pair of colons (including any option symbols) in front of its name. For example: `Gosub ::xyz`. However, jumping to a [single-line \(auto-replace\) hotstring](#) will do nothing other than execute a [return](#).

Although hotstrings are not monitored and will not be triggered during the course of an invisible [Input](#) command, visible Inputs are capable of triggering them.

By default, hotstrings are never triggered by keystrokes produced by any AutoHotkey script. This avoids the possibility of an infinite loop where

hotstrings trigger each other over and over. This behaviour can be controlled with `#InputLevel` and `SendLevel`. However, auto-replace hotstrings always use send level 0 and therefore never trigger `hook hotkeys` or hotstrings.

The `Input` command is more flexible than hotstrings for certain purposes. For example, it allows your keystrokes to be invisible in the active window (such as a game). It also supports non-character ending keys such as Escape.

The `keyboard hook` is automatically used by any script that contains hotstrings.

Hotstrings behave identically to hotkeys in the following ways:

- They are affected by the `Suspend` command.
- They obey `#MaxThreads` and `#MaxThreadsPerHotkey` (but not `#MaxThreadsBuffer`).
- Scripts containing hotstrings are automatically `persistent`.
- Non-auto-replace hotstrings will create a new `thread` when launched. In addition, they will update the built-in hotkey variables such as `A_ThisHotkey`.

Known limitation: On some systems in Java applications, hotstrings might interfere with the user's ability to type diacritical letters (via dead keys). To work around this, `Suspend` can be turned on temporarily (which disables all hotstrings).

Hotstring Helper

Andreas Borutta suggested the following script, which might be useful if you are a heavy user of hotstrings. By pressing Win+H (or another hotkey of your choice), the currently selected text can be turned into a hotstring. For example, if you have "by the way" selected in a word processor, pressing Win+H will prompt you for its abbreviation (e.g. btw) and then add the new hotstring to the script. It will then reload the script to activate the hotstring.

```
#h:: ; Win+H hotkey
; Get the text currently selected. The clipboard
is used instead of
; "ControlGetSelected" because it works in a
greater variety of editors
; (namely word processors). Save the current
clipboard contents to be
; restored later. Although this handles only plain
text, it seems better
; than nothing:
ClipboardOld := ClipboardAll
Clipboard := "" ; Must start off blank for
detection to work.
Send ^c
ClipWait 1
if ErrorLevel ; ClipWait timed out.
    return
; Replace CRLF and/or LF with `n for use in a
"send-raw" hotstring:
; The same is done for any other characters that
might otherwise
; be a problem in raw mode:
StrReplace, Hotstring, % Clipboard, `n, `r`n ; Do
```

this replacement first to avoid interfering with the others below.

```
StrReplace, Hotstring, % Hotstring, `r`n, ``r ;
```

Using `r works better than `n in MS Word, etc.

```
StrReplace, Hotstring, % Hotstring, `n, ``r
```

```
StrReplace, Hotstring, % Hotstring, % A_Tab, ``t
```

```
StrReplace, Hotstring, % Hotstring, `;`,`;`;
```

```
Clipboard := ClipboardOld ; Restore previous contents of clipboard.
```

```
; This will move the InputBox's caret to a more friendly position:
```

```
SetTimer, MoveCaret, 10
```

```
; Show the InputBox, providing the default hotstring:
```

```
InputBox, Hotstring, Type your abbreviation at the indicated insertion point. You can also edit the replacement text if you wish.`n`nExample entry:
```

```
:R:btw`:::by the way, New Hotstring, ,
```

```
:R`:::%Hotstring%
```

```
if ErrorLevel ; The user pressed Cancel.
```

```
return
```

```
if InStr(Hotstring,":R`:::")
```

```
{
```

```
MsgBox You didn't provide an abbreviation. The hotstring has not been added.
```

```
return
```

```
}
```

```
; Otherwise, add the hotstring and reload the script:
```

```
FileAppend, `n%Hotstring%, %A_ScriptFullPath% ;
```

```
Put a `n at the beginning in case file lacks a blank line at its end.
```

```
Reload
```

```
Sleep 200 ; If successful, the reload will close this instance during the Sleep, so the line below will never be reached.
```

```
Result := MsgBox("The hotstring just added appears
```

```
to be improperly formatted. Would you like to
open the script for editing? Note that the bad
hotstring is at the bottom of the script.",, 4)
if Result = "Yes"
    Edit
return
```

```
MoveCaret:
if !WinActive("New Hotstring")
    return
```

```
; Otherwise, move the InputBox's insertion point
to where the user will type the abbreviation.
```

```
Send {Home}{Right 3}
SetTimer, MoveCaret, Off
return
```

Remapping Keys and Buttons

Introduction

Limitation: AutoHotkey's remapping feature described below is generally not as pure and effective as remapping directly via the Windows registry. For the advantages and disadvantages of each approach, see [registry remapping](#).

Remapping the Keyboard and Mouse

The syntax for the built-in remapping feature is

`OriginKey::DestinationKey`. For example, a [script](#) consisting only of the following line would make the "a" key behave like the "b" key:

```
a::b
```

The above example does not alter the "b" key itself. The "b" key would continue to send the "b" keystroke unless you remap it to something else as shown in the following example:

```
a::b  
b::a
```

The examples above use lowercase, which is recommended for most purposes because it also remaps the corresponding uppercase letters (that is, it will send uppercase when Capslock is "on" or the Shift key is held down). By contrast, specifying an uppercase letter on the right side forces uppercase. For example, the following line would produce an uppercase B when you type either "a" or "A" (as long as Capslock is off):

```
a::B
```

Mouse remapping: To remap the mouse instead of the keyboard, use the same

approach. For example:

MButton::Shift	Makes the middle button behave like the Shift key.
XButton1::LButton	Makes the fourth mouse button behave like the left mouse button.
RAlt::RButton	Makes the right Alt key behave like the right mouse button.

Other useful remappings:

Capslock::Ctrl	Makes Capslock become a Control key. To retain the ability to turn Capslock on and off, add the remapping <code>+Capslock::Capslock</code> first. This toggles Capslock on and off when you hold down the Shift key and press Capslock. Because both remappings allow additional modifier keys to be held down, the more specific <code>+Capslock::Capslock</code> remapping must be placed first for it to work.
XButton2::^LButton	Makes the fifth mouse button (XButton2) produce Control-LeftClick.
RAlt::AppsKey	Makes the right Alt key become the Apps key (which is the key that opens the context menu).
RCtrl::RWin	Makes the right Control key become the right Windows key.
Ctrl::Alt	Makes both Control keys behave like an Alt key. However, see alt-tab issues .
^X::^C	Makes Control-X produce Control-C. It also makes Control-Alt-X produce Control-Alt-C, etc.
RWin::Return	Disables the right Windows key by having it simply return .

You can try out any of these examples by copying them into a new text file such as "Remap.ahk", then launching the file.

See the [Key List](#) for a complete list of key and mouse button names.

Remarks

The directives `#IfWinActive/Exist` can be used to make selected remappings active only in the windows you specify. For example:

```
#IfWinActive ahk_class Notepad
a::b ; Makes the 'a' key send a 'b' key, but
only in Notepad.
#IfWinActive ; This puts subsequent remappings
and hotkeys in effect for all windows.
```

Remapping a key or button is "complete" in the following respects:

- Holding down a modifier such as Control or Shift while typing the origin key will put that modifier into effect for the destination key. For example, `b::a` would produce Control-A if you press Control-B.
- Capslock generally affects remapped keys in the same way as normal keys.
- The destination key or button is held down for as long as you continue to hold down the origin key. However, some games do not support remapping; in such cases, the keyboard and mouse will behave as though not remapped.
- Remapped keys will auto-repeat while being held down (except keys remapped to become mouse buttons).

Although a remapped key can trigger normal hotkeys, by default it cannot trigger mouse hotkeys or [hook hotkeys](#) (use [ListHotkeys](#) to discover which hotkeys are "hook"). For example, if the remapping `a::b` is in effect, pressing Ctrl-Alt-A would trigger the `^!b` hotkey only if `^!b` is not a hook hotkey. If

`^!b` is a hook hotkey, you can define `^!a` as a hotkey if you want Ctrl-Alt-A to perform the same action as Ctrl-Alt-B. For example:

```
a::b
^!a::
^!b::
ToolTip You pressed %A_ThisHotkey%.
return
```

Alternatively, `#InputLevel` can be used to override the default behaviour. For example:

```
#InputLevel 1
a::b

#InputLevel 0
^!b::
ToolTip You pressed %A_ThisHotkey%.
return
```

If `SendMode` is used in the auto-execute section (top part of the script), it affects all remappings. However, since remapping uses `Send {Blind}` and since the `SendPlay` mode does not fully support `{Blind}`, some remappings might not function properly in `SendPlay` mode (especially Control, Shift, Alt, and Win). To work around this, avoid `SendPlay` in auto-execute section when you have remappings; then use the command `SendPlay` vs. `Send` in other places throughout the script. Alternatively, you could translate your remappings into hotkeys (as described below) that explicitly call `SendEvent` vs. `Send`.

When a script is launched, each remapping is translated into a pair of **hotkeys**. For example, a script containing `a::b` actually contains the following two hotkeys instead:

```
*a::
SetKeyDelay -1 ; If the destination key is a
mouse button, SetMouseDelay is used instead.
Send {Blind}{b DownTemp} ; DownTemp is like
Down except that other Send commands in the
script won't assume "b" should stay down during
their Send.
return

*a up::
SetKeyDelay -1 ; See note below for why press-
duration is not specified with either of these
SetKeyDelays.
Send {Blind}{b Up}
return
```

However, the above hotkeys vary under the following circumstances:

1. When the source key is LCtrl and the destination key is an Alt key, the line `Send {Blind}{LAlt DownTemp}` is replaced by `Send {Blind}{LCtrl Up}{LAlt DownTemp}`. The same is true if the source is RCtrl, except that `{RCtrl up}` is used.
2. When a keyboard key is being remapped to become a mouse button (e.g. `RCtrl::RButton`), the hotkeys above use SetMouseDelay in place of SetKeyDelay. In addition, the first hotkey above is replaced by the following, which prevents the keyboard's auto-repeat feature from

generating repeated mouse clicks:

```
*RCtrl::  
SetMouseDelay -1  
if not GetKeyState("RButton") ; i.e. the  
right mouse button isn't down yet.  
    Send {Blind}{RButton DownTemp}  
return
```

Note that SetKeyDelay's second parameter ([press duration](#)) is omitted in the hotkeys above. This is because press-duration does not apply to down-only or up-only events such as `{b down}` and `{b up}`. However, it does apply to changes in the state of the Shift/Ctrl/Alt/Win keys, which affects remappings such as `a::E` or `a::^b`. Consequently, any press-duration a script puts into effect via its [auto-execute section](#) will apply to all such remappings.

Although a pair of keys cannot be directly remapped to single key (e.g. it's invalid to write `a & c::b`), this effect can be achieved by explicitly adding the up and down hotkeys from the example higher above: simply replace `*a::` with `a & c::`, and replace `*a up::` with `a & c up::`.

Since remappings are translated into hotkeys as described above, the [Suspend](#) command affects them. Similarly, the [Hotkey](#) command can disable or modify a remapping. For example, the following two commands would disable the remapping `a::b`.

```
Hotkey, *a, off  
Hotkey, *a up, off
```

Alt-tab issues: If you remap a key or mouse button to become an Alt key, that key will probably not be able to alt-tab properly. A possible work-around is to add the hotkey `*Tab::Send {Blind}{Tab}` -- but be aware that it will likely interfere with using the real Alt key to alt-tab. Therefore, it should be used only when you alt-tab solely by means of remapped keys and/or [alt-tab hotkeys](#).

In addition to the keys and mouse buttons on the [Key List](#) page, the source key may also be a virtual key (VKnn) or scan code (SCnnn) as described on the [special keys](#) page. The same is true for the destination key except that it may optionally specify a scan code after the virtual key. For example, `sc01e::vk42sc030` is equivalent to `a::b` on most keyboard layouts.

To disable a key rather than remapping it, make it a hotkey that simply [returns](#). For example, `F1::return` would disable the F1 key.

The following keys are not supported by the built-in remapping method:

- The mouse wheel (WheelUp/Down/Left/Right).
- Pause and Break as destination keys (since they match the names of commands).
- Curly braces {} as destination keys. Instead use the [VK/SC method](#); e.g. `x::+sc01A` and `y::+sc01B`.
- A percent sign (%) as a destination key. Instead use the [VK/SC method](#).
- "Return" as a destination key. Instead use "Enter".

Moving the Mouse Cursor via the Keyboard

The keyboard can be used to move the mouse cursor as demonstrated by the fully-featured [Keyboard-To-Mouse script](#). Since that script offers smooth cursor movement, acceleration, and other features, it is the recommended approach if you plan to do a lot of mousing with the keyboard. By contrast, the following example is a simpler demonstration:

```
*#up::MouseMove, 0, -10, 0, R ; Win+UpArrow
hotkey => Move cursor upward
*#Down::MouseMove, 0, 10, 0, R ; Win+DownArrow
=> Move cursor downward
*#Left::MouseMove, -10, 0, 0, R ;
Win+LeftArrow => Move cursor to the left
*#Right::MouseMove, 10, 0, 0, R ;
Win+RightArrow => Move cursor to the right

*<#RCtrl:: ; LeftWin + RightControl => Left-
click (hold down Control/Shift to Control-Click
or Shift-Click).
SendEvent {Blind}{LButton down}
KeyWait RCtrl ; Prevents keyboard auto-repeat
from repeating the mouse click.
SendEvent {Blind}{LButton up}
return

*<#AppsKey:: ; LeftWin + AppsKey => Right-
click
SendEvent {Blind}{RButton down}
KeyWait AppsKey ; Prevents keyboard auto-
repeat from repeating the mouse click.
SendEvent {Blind}{RButton up}
return
```


Remapping via the Registry's "Scancode Map"

Advantages:

- Registry remapping is generally more pure and effective than [AutoHotkey's remapping](#). For example, it works in a broader variety of games, it has no known [alt-tab issues](#), and it is capable of firing AutoHotkey's hook hotkeys (whereas AutoHotkey's remapping requires a [workaround](#)).
- If you choose to make the registry entries manually (explained below), absolutely no external software is needed to remap your keyboard. Even if you use [KeyTweak](#) to make the registry entries for you, KeyTweak does not need to stay running all the time (unlike AutoHotkey).

Disadvantages:

- Registry remapping is relatively permanent: a reboot is required to undo the changes or put new ones into effect.
- Its effect is global: it cannot create remappings specific to a particular user, application, or locale.
- It cannot send keystrokes that are modified by Shift, Control, Alt, or AltGr. For example, it cannot remap a lowercase character to an uppercase one.
- It supports only the keyboard (AutoHotkey has [mouse remapping](#) and some [limited joystick remapping](#)).

How to Apply Changes to the Registry: There are at least two methods to remap keys via the registry:

1. Use a program like [KeyTweak](#) (freeware) to visually remap your keys. It will change the registry for you.
2. Remap keys manually by creating a .reg file (plain text) and loading it into the registry. This is demonstrated at www.autohotkey.com/forums/post-56216.html#56216

Related Topics

[List of keys and mouse buttons](#) [GetKeyState](#)

[Remapping a joystick](#)

List of Keys, Mouse Buttons, and Joystick Controls

Mouse

General	
LButton	Left mouse button
RButton	Right mouse button
MButton	Middle or wheel mouse button
Advanced	
XButton1	4th mouse button. Typically performs the same function as <code>Browser_Back</code> .
XButton2	5th mouse button. Typically performs the same function as <code>Browser_Forward</code> .
Wheel	
WheelDown	Turn the wheel downward (toward you).
WheelUp	Turn the wheel upward (away from you).
WheelLeft WheelRight	Scroll to the left or right. Requires Windows Vista or later. These can be used as hotkeys with some (but not all) mice which have a second wheel or support tilting the wheel to either side. In some cases, software bundled with the mouse must instead be used to control this feature. Regardless of the particular mouse, Send and Click can be used to scroll horizontally in programs which support it.

Keyboard

Note: The names of the letter and number keys are the same as that single letter or digit. For example: b is the "b" key and 5 is the "5" key.

General	
CapsLock	Caps lock
Space	Space bar
Tab	Tab key
Enter (or Return)	Enter key
Escape (or Esc)	Esc key
Backspace (or BS)	Backspace
Cursor Control	
ScrollLock	Scroll lock
Delete (or Del)	Delete key
Insert (or Ins)	Insert key
Home	Home key
End	End key
PgUp	Page Up key
PgDn	Page Down key
Up	Up arrow key
Down	Down arrow key
Left	Left arrow key
Right	Right arrow key
Numpad	

Due to system behavior, the following keys are identified differently depending on whether NumLock is ON or OFF. If NumLock is OFF but Shift is pressed, the system temporarily releases Shift and acts as though NumLock is ON.

NumLock ON	NumLock OFF	
Numpad0	NumpadIns	0 / Insert key
Numpad1	NumpadEnd	1 / End key
Numpad2	NumpadDown	2 / Down arrow key
Numpad3	NumpadPgDn	3 / Page Down key
Numpad4	NumpadLeft	4 / Left arrow key
Numpad5	NumpadClear	5 / typically does nothing
Numpad6	NumpadRight	6 / Right arrow key
Numpad7	NumpadHome	7 / Home key
Numpad8	NumpadUp	8 / Up arrow key
Numpad9	NumpadPgUp	9 / Page Up key
NumpadDot	NumpadDel	Decimal separation / Delete key
Not affected by NumLock		
NumLock	Number lock	
NumpadDiv	Divide	
NumpadMult	Multiply	
NumpadAdd	Add	
NumpadSub	Subtract	
NumpadEnter	Enter key	
Function		
F1 - F24	The 12 or more function keys at the top of most keyboards.	

Modifier	
LWin	Left Windows logo key. Corresponds to the <code><#</code> hotkey prefix.
RWin	Right Windows logo key. Corresponds to the <code>>#</code> hotkey prefix. Note: Unlike Control/Alt/Shift, there is no generic/neutral "Win" key because the OS does not support it. However, hotkeys with the <code>#</code> modifier can be triggered by either Win key.
Control (or Ctrl)	Control key. As a hotkey (<code>Control::</code>) it fires upon release unless it has the tilde prefix. Corresponds to the <code>^</code> hotkey prefix.
Alt	Alt key. As a hotkey (<code>Alt::</code>) it fires upon release unless it has the tilde prefix. Corresponds to the <code>!</code> hotkey prefix.
Shift	Shift key. As a hotkey (<code>Shift::</code>) it fires upon release unless it has the tilde prefix. Corresponds to the <code>+</code> hotkey prefix.
LControl (or LCtrl)	Left Control key. Corresponds to the <code><^</code> hotkey prefix.
RControl (or RCtrl)	Right Control key. Corresponds to the <code>>^</code> hotkey prefix.
LShift	Left Shift key. Corresponds to the <code><+</code> hotkey prefix.
RShift	Right Shift key. Corresponds to the <code>>+</code> hotkey prefix.
LAlt	Left Alt key. Corresponds to the <code><!</code> hotkey prefix.
RAlt	Right Alt key. Corresponds to the <code>>!</code> hotkey prefix. Note: If your keyboard layout has AltGr instead of RAlt, you can probably use it as a hotkey prefix via <code><^>!</code> as described here . In addition, <code>LControl & RAlt::</code> would make AltGr itself into a hotkey.
Multimedia	

Browser_Back	Back
Browser_Forward	Forward
Browser_Refresh	Refresh
Browser_Stop	Stop
Browser_Search	Search
Browser_Favorites	Favorites
Browser_Home	Homepage
Volume_Mute	Mute the volume
Volume_Down	Lower the volume
Volume_Up	Increase the volume
Media_Next	Next Track
Media_Prev	Previous Track
Media_Stop	Stop
Media_Play_Pause	Play/Pause
Launch_Mail	Launch default e-mail program
Launch_Media	Launch default media player
Launch_App1	Launch My Computer
Launch_App2	Launch Calculator

Note: The function assigned to each of the keys listed above can be overridden by modifying the Windows registry. This table shows the default function of each key on most versions of Windows.

Special	
AppsKey	Menu key. This is the key that invokes the right-click context menu.
PrintScreen	Print screen
CtrlBreak	

Pause	Pause key
Break	Break key. Since this is synonymous with Pause, use <code>^CtrlBreak</code> in hotkeys instead of <code>^Pause</code> or <code>^Break</code> .
Help	Help key. This probably doesn't exist on most keyboards. It's usually not the same as F1.
Sleep	Sleep key. Note that the sleep key on some keyboards might not work with this.
SCnnn	Specify for nnn the scan code of a key. Recognizes unusual keys not mentioned above. See Special Keys for details.
VKnn	<p>Specify for nn the hexadecimal virtual key code of a key. This rarely-used method also prevents certain types of hotkeys from requiring the keyboard hook. For example, the following hotkey does not use the keyboard hook, but as a side-effect it is triggered by pressing <i>either</i> Home or NumpadHome:</p> <pre style="background-color: #ffffcc; padding: 10px; border: 1px solid #ccc;">^VK24::MsgBox You pressed Home or NumpadHome while holding down Control.</pre> <p>Known limitation: VK hotkeys that are forced to use the keyboard hook, such as <code>^VK24</code> or <code>~VK24</code>, will fire for only one of the keys, not both (e.g. NumpadHome but not Home). For more information about the VKnn method, see Special Keys.</p>

Joystick

Joy1 through Joy32: The buttons of the joystick. To help determine the button numbers for your joystick, use this [test script](#). Note that [hotkey prefix symbols](#) such as ^ (control) and + (shift) are not supported (though [GetKeyState](#) can be used as a substitute). Also note that the pressing of joystick buttons always "passes through" to the active window if that window is designed to detect the pressing of joystick buttons.

Although the following Joystick control names cannot be used as hotkeys, they can be used with [GetKeyState](#):

JoyX, JoyY, and JoyZ: The X (horizontal), Y (vertical), and Z (altitude/depth) axes of the joystick.

JoyR: The rudder or 4th axis of the joystick.

JoyU and JoyV: The 5th and 6th axes of the joystick.

JoyPOV: The point-of-view (hat) control.

JoyName: The name of the joystick or its driver.

JoyButtons: The number of buttons supported by the joystick (not always accurate).

JoyAxes: The number of axes supported by the joystick.

JoyInfo: Provides a string consisting of zero or more of the following letters to indicate the joystick's capabilities: **Z** (has Z axis), **R** (has R axis), **U** (has U axis), **V** (has V axis), **P** (has POV control), **D** (the POV control has a limited number of discrete/distinct settings), **C** (the POV control is continuous/fine). Example string: ZRUVPD

Multiple Joysticks: If the computer has more than one joystick and you want to use one beyond the first, include the joystick number (max 16) in front of the control name. For example, 2joy1 is the second joystick's first button.

Note: If you have trouble getting a script to recognize your joystick, one person reported needing to specify a joystick number other than 1 even though only a single joystick was present. It is unclear how this situation arises or whether it is normal, but experimenting with the joystick number in the [joystick test script](#) can help determine if this applies to your system.

See Also:

[Joystick remapping:](#) methods of sending keystrokes and mouse clicks with a joystick.

[Joystick-To-Mouse script:](#) using a joystick as a mouse.

Hand-held Remote Controls

Respond to signals from hand-held remote controls via the [WinLIRC client script](#).

Special Keys

If your keyboard or mouse has a key not listed above, you might still be able to make it a hotkey by using the following steps:

1. Ensure that at least one script is running that is using the [keyboard hook](#). You can tell if a script has the keyboard hook by opening its main window and selecting "View->Key history" from the menu bar.
2. Double-click that script's tray icon to open its main window.
3. Press one of the "mystery keys" on your keyboard.
4. Select the menu item "View->Key history"
5. Scroll down to the bottom of the page. Somewhere near the bottom are the key-down and key-up events for your key. NOTE: Some keys do not generate events and thus will not be visible here. If this is the case, you cannot directly make that particular key a hotkey because your keyboard driver or hardware handles it at a level too low for AutoHotkey to access. For possible solutions, see further below.
6. If your key is detectible, make a note of the 3-digit hexadecimal value in the second column of the list (e.g. **159**).
7. To define this key as a hotkey, follow this example:

```
SC159:: ; Replace 159 with your key's value.  
MsgBox, %A_ThisHotKey% was pressed.  
return
```

Reverse direction: To remap some other key to *become* a "mystery key", follow

this example:

```
; Replace 159 with the value discovered above.  
Replace FF (if needed) with the  
; key's virtual key, which can be discovered in  
the first column of the Key History screen.  
#C::Send {vkFFsc159}
```

Alternate solutions: If your key or mouse button is not detectible by the Key History screen, one of the following might help:

1. Reconfigure the software that came with your mouse or keyboard (sometimes accessible in the Control Panel or Start Menu) to have the "mystery key" send some other keystroke. Such a keystroke can then be defined as a hotkey in a script. For example, if you configure a mystery key to send Control+F1, you can then indirectly make that key as a hotkey by using `^F1::` in a script.

2. Try [AHKHID](#). You can also try searching the [forum](#) for a keywords like `RawInput *`, `USB HID` or `AHKHID`.

3. The following is a last resort and generally should be attempted only in desperation. This is because the chance of success is low and it may cause unwanted side-effects that are difficult to undo:

Disable or remove any extra software that came with your keyboard or mouse or change its driver to a more standard one such as the one built into the OS. This assumes there is such a driver for your particular keyboard or mouse and that you can live without the features provided by its custom

driver and software.

Scripts

Table of Contents

- [Introduction](#)
- [The Top of the Script \(the Auto-execute Section\)](#): This portion executes automatically when the script starts.
- [Escape Sequences](#): When to use ``%` and ```, to indicate a literal percent sign or comma.
- [Comments in Scripts](#): The use of semicolon and the symbols `/*...*/` to add remarks to a script.
- [Splitting a Long Line into a Series of Shorter Ones](#): This can improve a script's readability and maintainability.
- [Convert a Script to an EXE \(ahk2exe\)](#): Convert a `.ahk` script into a `.exe` file that can run on any PC.
- [Passing Command Line Parameters to a Script](#): The variable `A_Args` contains the incoming parameters.
- [Script File Codepage](#): Using non-ASCII characters safely in scripts.
- [Debugging a Script](#): How to find the flaws in a misbehaving script.
- [Portability of AutoHotkey.exe](#): Having a copy of `AutoHotkey.exe` is enough to execute any `.ahk` file.
- [Installer Options](#): How to do unattended/silent installations or uninstallations.

Introduction

Each script is a plain text file containing lines to be executed by the program (AutoHotkey.exe). A script may also contain [hotkeys](#) and [hotstrings](#), or even consist entirely of them. However, in the absence of hotkeys and hotstrings, a script will perform its commands sequentially from top to bottom the moment it is launched.

The program loads the script into memory line by line, and each line may be up to 16,383 characters long. During loading, the script is [optimized](#) and validated. Any syntax errors will be displayed, and they must be corrected before the script can run.

The Top of the Script (the Auto-execute Section)

After the script has been loaded, it begins executing at the top line, continuing until a [Return](#), [Exit](#), [hotkey/hotstring label](#), or the physical end of the script is encountered (whichever comes first). This top portion of the script is referred to as the *auto-execute* section.

A script that lacks [hotkeys](#), [hotstrings](#), [visible GUIs](#), [active message monitors](#), [active timers](#), [active OnClipboardChange](#) callback functions, [custom tray menu items](#) and the [#Persistent](#) directive will terminate after the auto-execute section has completed. Otherwise, it will stay running in an idle state, responding to events such as [hotkeys](#), [hotstrings](#), [GUI events](#), [custom menu items](#), and [timers](#). If these conditions change after the auto-execute section completes (for example, the last timer is disabled), the script may exit when the last running thread completes or the last GUI closes.

Every [thread](#) launched by a [hotkey](#), [hotstring](#), [menu item](#), [GUI event](#), or [timer](#) starts off fresh with the default values for the following attributes as set in the auto-execute section. If unset, the standard defaults will apply (as documented on each of the following pages): [DetectHiddenWindows](#), [DetectHiddenText](#), [SetTitleMatchMode](#), [SendMode](#), [SetKeyDelay](#), [SetMouseDelay](#), [SetWinDelay](#), [SetControlDelay](#), [SetDefaultMouseSpeed](#), [CoordMode](#), [SetStoreCapslockMode](#), [StringCaseSense](#), [Thread](#), and [Critical](#).

If the auto-execute section takes a long time to complete (or never completes), the default values for the above settings will be put into effect after 100

milliseconds. When the auto-execute section finally completes (if ever), the defaults are updated again to be those that were in effect at the end of the auto-execute section. Thus, it's usually best to make any desired changes to the defaults at the top of scripts that contain [hotkeys](#), [hotstrings](#), [timers](#), or [custom menu items](#). Also note that each [thread](#) retains its own collection of the above settings. Changes made to those settings will not affect other [threads](#).

Escape Sequences

AutoHotkey's [escape character](#) is accent/backtick (`), which is at the upper left corner of most English keyboards. Using this character rather than backslash avoids the need for double backslashes in file paths.

A quote mark can be escaped to include it inside a [quoted string](#). For example, the expressions `"`"`, "`"`, and Chr(34) all produce a string containing a double quote mark.`

Certain special characters are also produced by means of an escape sequence. The most common ones are ``t` (tab), ``n` (linefeed), and ``r` (carriage return).

Comments in Scripts

Scripts can be commented by using a semicolon at the beginning of a line. For example:

```
; This entire line is a comment.
```

Comments may also be added to the end of a command, in which case the semicolon must have at least one space or tab to its left. For example:

```
Run "Notepad" ; This is a comment on the same  
line as a command.
```

In addition, the `/*` and `*/` symbols can be used to comment out an entire section, as in this example:

```
/*  
MsgBox "This line is commented out (disabled)."  
MsgBox "This one too."  
*/
```

Excluding whitespace, `/*` must appear at the start of the line, while `*/` can appear only at the start or end of a line. It is also valid to omit `*/`, in which case the remainder of the file is commented out.

Since comments are ignored when a script is launched, they do not impact performance or memory utilization.

Splitting a Long Line into a Series of Shorter Ones

Long lines can be divided up into a collection of smaller ones to improve readability and maintainability. This does not reduce the script's execution speed because such lines are merged in memory the moment the script launches.

Method #1: A line that starts with "and", "or", ||, &&, a comma, or a period is automatically merged with the line directly above it (the same is true for all other expression operators except ++ and --). In the following example, the second line is appended to the first because it begins with a comma:

```
FileAppend "This is the text to append.`n" ;  
A comment is allowed here.  
    , A_ProgramFiles  
    "\SomeApplication\LogFile.txt" ; Comment.
```

Similarly, the following lines would get merged into a single line because the last two start with "and" or "or":

```
if (Color = "Red" or Color = "Green" or Color  
= "Blue" ; Comment.  
    or Color = "Black" or Color = "Gray" or  
Color = "White") ; Comment.  
    and ProductIsAvailableInColor(Product,  
Color) ; Comment.
```

The ternary operator is also a good candidate:

```
ProductIsAvailable := (Color = "Red")
```

```
? false ; We don't have any red products,  
so don't bother calling the function.  
: ProductIsAvailableInColor(Product, Color)
```

Although the indentation used in the examples above is optional, it might improve clarity by indicating which lines belong to ones above them. Also, it is not necessary to include extra spaces for lines starting with the words "AND" and "OR"; the program does this automatically. Finally, blank lines or [comments](#) may be added between or at the end of any of the lines in the above examples.

Method #2: This method should be used to merge a large number of lines or when the lines are not suitable for Method #1. Although this method is especially useful for [auto-replace hotstrings](#), it can also be used with any command or [expression](#). For example:

```
; EXAMPLE #1:  
Var := "  
(  
Line 1 of the text.  
Line 2 of the text. By default, a linefeed (`n)  
is present between lines.  
)"  
  
; EXAMPLE #2:  
FileAppend "  
(  
A line of text.  
By default, the hard carriage return (Enter)  
between the previous line and this one will be  
written to the file.  
    This line is indented with a tab; by  
default, that tab will also be written to the
```

```
file.  
)", A_Desktop "\My File.txt"
```

In the examples above, a series of lines is bounded at the top and bottom by a pair of parentheses. This is known as a *continuation section*. Notice that the bottom line contains `FileAppend`'s last parameter after the closing parenthesis. This practice is optional; it is done in cases like this so that the comma will be seen as a parameter-delimiter rather than a literal comma.

By default, quote marks inside the continuation section act as if they have been escaped (i.e. they are interpreted as literal characters). Additionally, leading spaces or tabs are omitted based on the indentation of the first line inside the continuation section. If the first line mixes spaces and tabs, only the first type of character is treated as indentation. If any line is indented less than the first line or with the wrong characters, all leading whitespace on that line is left as is.

The default behavior of a continuation section can be overridden by including one or more of the following options to the right of the section's opening parenthesis. If more than one option is present, separate each one from the previous with a space. For example: `(LTrim Join|`.

Join: Specifies how lines should be connected together. If this option is omitted, each line except the last will be followed by a linefeed character (`\n`). If the word *Join* is specified by itself, lines are connected directly to each other without any characters in between. Otherwise, the word *Join* should be followed immediately by as many as 15 characters. For example, `Join`s` would insert a space after each line except the last (`"`s"` indicates a literal space -- it is a special escape

sequence recognized only by *Join*). Another example is `Join`r`n`, which inserts CR+LF between lines. Similarly, `Join|` inserts a pipe between lines. To have the final line in the section also ended by a join-string, include a blank line immediately above the section's closing parenthesis.

Known limitation: If the Join string ends with a colon, it must not be the last option on the line. For example, `(Join:` is treated as the label "(Join" and `(LTrim Join:` is unsupported, but `(Join: C` is okay.

LTrim: Omits all spaces and tabs at the beginning of each line. This is usually unnecessary because of the default "smart" behaviour.

LTrim0 (LTrim followed by a zero): Turns off the omission of spaces and tabs from the beginning of each line.

RTrim0 (RTrim followed by a zero): Turns off the omission of spaces and tabs from the end of each line.

Comments (or **Comment** or **Com** or **C**): Allows [semicolon comments](#) inside the continuation section (but not `/*. . */`). Such comments (along with any spaces and tabs to their left) are entirely omitted from the joined result rather than being treated as literal text. Each comment can appear to the right of a line or on a new line by itself.

Quotes (or **Q**): Restores the ability to terminate quoted strings when an expression is used in a continuation section.

``` (accent): Treats each backtick character literally rather than as an [escape](#)

character. This also prevents commas and percent signs from being explicitly and individually escaped. In addition, it prevents the translation of any explicitly specified escape sequences such as ``r` and ``t`.

): If a closing parenthesis appears in the continuation section's options (except as a parameter of the `Join` option), the line is reinterpreted as an expression instead of the beginning of a continuation section. This allows expressions like `(x.y)` `[z]()` to work without the need to escape the opening parenthesis.

Escape sequences such as ``n` (linefeed) and ``t` (tab) are supported inside the continuation section except when the `accent`()` option has been specified.

When the `comment` option is absent, semicolon and `/*..*/` comments are not supported within the interior of a continuation section because they are seen as literal text. However, comments can be included on the bottom and top lines of the section. For example:

```
FileAppend " ; Comment.
; Comment.
(LTrim Join ; Comment.
 ; This is not a comment; it is literal.
Include the word Comments in the line above to
make it a comment.
)", "C:\File.txt" ; Comment.
```

As a consequence of the above, semicolons never need to be escaped within a continuation section.

A continuation section cannot produce a line whose total length is greater than

16,383 characters (if it tries, the program will alert you the moment the script is launched). One way to work around this is to do a series of concatenations into a variable. For example:

```
Var := "
(
...
)"
Var := "`n "; Add more text to the variable via
another continuation section.
(
...
)"
FileAppend Var, "C:\My File.txt"
```

Since a closing parenthesis indicates the end of a continuation section, to have a line start with literal closing parenthesis, precede it with an accent/backtick: ``)`.

A continuation section can be immediately followed by a line containing the open-parenthesis of another continuation section. This allows the options mentioned above to be varied during the course of building a single line.

The piecemeal construction of a continuation section by means of `#Include` is not supported.

## Convert a Script to an EXE (ahk2exe)

A script compiler (courtesy of fins) is included with the program.

Once a script is compiled, it becomes a standalone executable; that is, AutoHotkey.exe is not required in order to run the script. The compilation process creates an executable file which contains the following: the AutoHotkey interpreter, the script, any files it [includes](#), and any files it has incorporated via the [FileInstall](#) command.

Ahk2Exe can be used in the following ways:

1. **GUI Interface:** Run the "Convert .ahk to .exe" item in the Start Menu.
2. **Right-click:** Within an open Explorer window, you can right-click any .ahk file and select "Compile Script" (only available if the script compiler option was chosen when AutoHotkey was installed). This creates an EXE file of the same base filename as the script, which appears after a short time in the same directory. Note: The EXE file is produced using the same custom icon, .bin file and use [MPRESS](#) setting that were last used by Method #1 above.
3. **Command Line:** The compiler can be run from the command line with the following parameters:

```
Ahk2Exe.exe /in MyScript.ahk [/out
MyScript.exe] [/icon MyIcon.ico] [/bin
AutoHotkeySC.bin] [/mpress 0or1]
```

For example:

```
Ahk2Exe.exe /in "MyScript.ahk" /icon
"MyIcon.ico"
```

Usage:

- Parameters containing spaces should be enclosed in double quotes.
- If the "out" file is omitted, the EXE will have the same base filename as the script itself.

Notes:

- Compiling does not typically improve the performance of a script.
- The commands `#NoTrayIcon` and `"Menu, Tray, ShowMainWindow"` affect the behavior of compiled scripts.
- Custom version info (as seen in Explorer's file-properties dialog) can be added to your compiled scripts by using a utility such as Resource Hacker (freeware) to edit the file "AutoHotkeySC.bin". This file is contained in the "Compiler" subfolder where AutoHotkey was installed. [Compile\\_AHK II](#) can be used to facilitate this process. The compiled script can be edited instead of AutoHotkeySC.bin.
- The method above can also be used to change existing icons or add new ones to all compiled scripts.
- The built-in variable `A_IsCompiled` contains 1 if the script is running in compiled form. Otherwise, it is blank.
- When parameters are passed to Ahk2Exe, a message indicating the success or failure of the compiling process is written to stdout. Although the

message will not appear at the command prompt, it can be "caught" by means such as redirecting output to a file.

- Additionally in the case of a failure, Ahk2Exe has exit codes indicating the kind of error that occurred. These error codes can be found at [GitHub \(ErrorCodes.md\)](#).
- Resources can be compressed to reduce size of the program.
- You can compile scripts with AutoHotkey.exe or AutoHotkey.dll.
- Main Script can be encrypted for protection, but you should recompile AutoHotkey\_H in Visual Studio with a different password or technique to have proper protection. Best is to use dynamic password generator or similar.
- WinApi functions are automatically declared for AutoHotkeySC.bin in compiled script using #DllImport.

The compiler's source code and newer versions can be found at [GitHub](#).

## Compressing Compiled Scripts

Ahk2Exe optionally uses MPRESS (a freeware program by MATCODE Software) to compress compiled scripts. If **mpress.exe** is present in the "Compiler" subfolder where AutoHotkey was installed, it is used automatically unless it is disabled via `/mpress 0` or the GUI setting.

Official website (was offline in March 2016):

<http://www.matcode.com/mpress.htm>

Mirror (downloads and information): <https://autohotkey.com/mpress/>

**Note:** While compressing the script executable prevents casual inspection of the script's source code using a plain text editor like Notepad or a PE resource editor, it does not prevent the source code from being extracted by tools dedicated to that purpose.

## Passing Command Line Parameters to a Script

Scripts support command line parameters. The format is:

```
AutoHotkey.exe [Switches] [Script Filename]
[Script Parameters]
```

And for compiled scripts, the format is:

```
CompiledScript.exe [Switches] [Script
Parameters]
```

**Switches:** Zero or more of the following:

| Switch                             | Meaning                                                                                                                                                                                                                     |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /f or /force                       | Launch unconditionally, skipping any warning dialogs. This has the same effect as <a href="#">#SingleInstance Off</a> .                                                                                                     |
| /r or /restart                     | Indicate that the script is being restarted (this is also used by the <a href="#">Reload</a> command, internally).                                                                                                          |
| /ErrorStdOut                       | Send syntax errors that prevent a script from launching to stderr rather than displaying a dialog. See <a href="#">#ErrorStdOut</a> for details. This can be combined with /iLib to validate the script without running it. |
| Not supported by compiled scripts: |                                                                                                                                                                                                                             |
| /Debug                             | Connect to a debugging client. For more details, see <a href="#">Interactive Debugging</a> .                                                                                                                                |
| /CPn                               | Overrides the default codepage used to read script files. For more details, see <a href="#">Script File Codepage</a> .                                                                                                      |
|                                    |                                                                                                                                                                                                                             |

AutoHotkey loads the script but does not run it. For each script file which is auto-included via [the library mechanism](#), two lines are written to the file specified by *OutFile*. These lines are written in the following format, where *LibDir* is the full path of the Lib folder and *LibFile* is the filename of the library:

/iLib  
"OutFile"

```
#Include LibDir\
#IncludeAgain LibDir\LibFile.ahk
```

If the output file exists, it is overwritten. *OutFile* can be `*` to write the output to stdout.

If the script contains syntax errors, the output file may be empty. The process exit code can be used to detect this condition; if there is a syntax error, the exit code is 2. The `/ErrorStdOut` switch can be used to suppress or capture the error message.

**Script Filename:** This can be omitted if there are no *Script Parameters*. If omitted (such as if you run AutoHotkey directly from the Start menu), the program looks for a script file called `AutoHotkey.ahk` in the following locations, in this order:

- The directory which contains the [AutoHotkey executable](#).
- The current user's [Documents](#) folder.

The filename `AutoHotkey.ahk` depends on the name of the executable used to run the script. For example, if you rename AutoHotkey.exe to MyScript.exe, it will attempt to find `MyScript.ahk`. If you run AutoHotkeyU32.exe without parameters, it will look for AutoHotkeyU32.ahk.

Specify an asterisk (\*) for the filename to read the script text from standard input (stdin). For an example, see [ExecScript\(\)](#).

**Script Parameters:** The string(s) you want to pass into the script, with each separated from the next by a space. Any parameter that contains spaces should be enclosed in quotation marks. A literal quotation mark may be passed in by preceding it with a backslash (\"). Consequently, any trailing slash in a quoted parameter (such as "C:\My Documents\") is treated as a literal quotation mark (that is, the script would receive the string C:\My Documents"). To remove such quotes, use `A_Args[1] := StrReplace(A_Args[1], '"')`

Incoming parameters, if present, are stored as an array in the built-in variable `A_Args`, and can be accessed using [array syntax](#). `A_Args[1]` contains the first parameter. The following example exits the script when too few parameters are passed to it:

```
if A_Args.Length() < 3
{
 MsgBox "This script requires at least 3
parameters but it only received " argc "."
 ExitApp
}
```

If the number of parameters passed into a script varies (perhaps due to the user dragging and dropping a set of files onto a script), the following example can be used to extract them one by one:

```
for n, param in A_Args ; For each parameter:
{
```

```
 MsgBox "Parameter number " n " is " param
 "."
}
```

If the parameters are file names, the following example can be used to convert them to their case-corrected long names (as stored in the file system), including complete/absolute path:

```
for n, GivenPath in A_Args ; For each
parameter (or file dropped onto a script):
{
 Loop Files, GivenPath, "FD" ; Include
files and directories.
 LongPath := A_LoopFileFullPath
 MsgBox "The case-corrected long path name
of file`n" GivenPath "`nis:`n" LongPath
}
```

Known limitation: dragging files onto a .ahk script may fail to work properly if 8-dot-3 (short) names have been turned off in an NTFS file system. One work-around is to [compile](#) the script then drag the files onto the resulting EXE.

## Script File Codepage

The characters a script file may contain are restricted by the codepage used to load the file.

- If the file begins with a UTF-8 or UTF-16 (LE) byte order mark, the appropriate codepage is used and the `/CPn` switch is ignored.
- If the `/CPn` switch is passed on the command-line, codepage `n` is used. For a list of valid numeric codepage identifiers, see [MSDN](#).
- In all other cases, the system default ANSI codepage is used.

Note that this applies only to script files loaded by AutoHotkey, not to file I/O within the script itself. `FileEncoding` controls the default encoding of files read or written by the script, while `IniRead` and `IniWrite` always deal in UTF-16 or ANSI.

As all text is converted (where necessary) to the [native string format](#), characters which are invalid or don't exist in the native codepage are replaced with a placeholder: '◆?'. This should only occur if there are encoding errors in the script file or the codepages used to save and load the file don't match.

`RegWrite` may be used to set the default for scripts launched from Explorer (e.g. by double-clicking a file):

```
; Uncomment the appropriate line below or leave
them all commented to
; reset to the default of the current build.
Modify as necessary:
```

```
; codepage := 0 ; System default ANSI
codepage
; codepage := 65001 ; UTF-8
; codepage := 1200 ; UTF-16
; codepage := 1252 ; ANSI Latin 1; Western
European (Windows)
if (codepage != "")
 codepage := " /CP" . codepage
cmd := '"' A_AhkPath '"' ' codepage ' "%1" %* '
key := "AutoHotkeyScript\Shell\Open\Command"
if A_IsAdmin ; Set for all users.
 RegWrite cmd, "REG_SZ", "HKCR\" key
else ; Set for current user only.
 RegWrite cmd, "REG_SZ",
 "HKCU\Software\Classes\" key
```

This assumes AutoHotkey has already been installed. Results may be less than ideal if it has not.

## Debugging a Script

Commands such as [ListVars](#) and [Pause](#) can help you debug a script. For example, the following two lines, when temporarily inserted at carefully chosen positions, create "break points" in the script:

```
ListVars
Pause
```

When the script encounters these two lines, it will display the current contents of all variables for your inspection. When you're ready to resume, un-pause the script via the File or Tray menu. The script will then continue until reaching the next "break point" (if any).

It is generally best to insert these "break points" at positions where the active window does not matter to the script, such as immediately before a `WinActivate` command. This allows the script to properly resume operation when you un-pause it.

The following commands are also useful for debugging: [ListLines](#), [KeyHistory](#), and [OutputDebug](#).

Some common errors, such as typos and missing "global" declarations, can be detected by [enabling warnings](#).

## Interactive Debugging

Interactive debugging is possible with a supported [DBGp client](#). Typically the following actions are possible:

- Set and remove breakpoints on lines - pause execution when a [breakpoint](#) is reached.
- Step through code line by line - step into, over or out of functions and subroutines.
- Inspect all variables or a specific variable.
- View the stack of running subroutines and functions.

Note that this functionality is disabled for compiled scripts.

To enable interactive debugging, first launch a supported debugger client then launch the script with the **/Debug** command-line switch.

```
AutoHotkey.exe /Debug[=SERVER:PORT] ...
```

*SERVER* and *PORT* may be omitted. For example, the following are equivalent:

```
AutoHotkey /Debug "myscript.ahk"
AutoHotkey /Debug=localhost:9000 "myscript.ahk"
```

To attach the debugger to a script which is already running, send it a message as shown below:

```
ScriptPath := "" ; SET THIS TO THE FULL PATH OF
THE SCRIPT
A_DetectHiddenWindows := true
if WinExist(ScriptPath " ahk_class AutoHotkey")
```

```
 ; Optional parameters:
 ; wParam = the IPv4 address of the
debugger client, as a 32-bit integer.
 ; lParam = the port which the debugger
client is listening on.
 PostMessage
DllCall("RegisterWindowMessage", "str",
"AHK_ATTACH_DEBUGGER")
```

Once the debugger client is connected, it may detach without terminating the script by sending the "detach" DBGp command.

## Portability of AutoHotkey.exe

The file AutoHotkey.exe is all that is needed to launch any .ahk script.

Renaming AutoHotkey.exe also changes which script it runs *by default*, which can be an alternative to compiling a script for use on a computer without AutoHotkey installed. For instance, *MyScript.exe* automatically runs *MyScript.ahk* if a filename is not supplied, but is also capable of running other scripts.

## Installer Options

To silently install AutoHotkey into the default directory (which is the same directory displayed by non-silent mode), pass the parameter `/S` to the installer. For example:

```
AutoHotkey110800_Install.exe /S
```

A directory other than the default may be specified via the `/D` parameter (in the absence of `/S`, this changes the default directory displayed by the installer). For example:

```
AutoHotkey110800_Install.exe /S /D=C:\Program
Files\AutoHotkey
```

**Version:** If AutoHotkey was previously installed, the installer automatically detects which version of AutoHotkey.exe to set as the default. Otherwise, the default is Unicode 32-bit or Unicode 64-bit depending on whether the OS is 64-bit. To override which version of AutoHotkey.exe is set as the default, pass one of the following switches:

- `/A32` or `/ANSI`: ANSI 32-bit.
- `/U64` or `/X64`: Unicode 64-bit (only valid on 64-bit systems).
- `/U32`: Unicode 32-bit.

For example, the following installs silently and sets ANSI 32-bit as the default:

```
AutoHotkey110800_Install.exe /S /A32
```

**Uninstall:** To silently uninstall AutoHotkey, pass the `/Uninstall` parameter to Installer.ahk. For example:

```
"C:\Program Files\AutoHotkey\AutoHotkey.exe"
"C:\Program Files\AutoHotkey\Installer.ahk"
/Uninstall
```

For AutoHotkey versions older than 1.1.08.00, use `uninst.exe /S`. For example:

```
"C:\Program Files\AutoHotkey\uninst.exe" /S
```

**Note:** Installer.ahk must be run as admin to work correctly.

**Extract:** Later versions of the installer include a link in the bottom-right corner to extract setup files without installing. If this function is present, the `/E` switch can be used to invoke it from the command line. For example:

```
AutoHotkey110903_Install.exe /D=F:\AutoHotkey
/E
```

**Restart scripts [v1.1.19.02+]:** In silent install/uninstall mode, running scripts are closed automatically, where necessary. Pass the `/R` switch to automatically reload these scripts using whichever EXE they were running on, **without** command line args. Setup will attempt to launch the scripts via Explorer, so they

do not run as administrator if UAC is enabled.

**Taskbar buttons** [v1.1.08+]: On Windows 7 and later, taskbar buttons for multiple scripts are automatically grouped together or combined into one button by default. The *Separate taskbar buttons* option disables this by registering each AutoHotkey executable as a **host app** (`IsHostApp`).

[v1.1.24.02+]: For command-line installations, specify `/IsHostApp` or `/IsHostApp=1` to enable the option and `/IsHostApp=0` to disable it.

## Run with UI Access [v1.1.24.02+]

The installer GUI has an option "Add 'Run with UI Access' to context menus". This context menu option provides a workaround for common [UAC-related issues](#) by allowing the script to automate administrative programs - without the script running as admin. To achieve this, the installer does the following:

- Copies AutoHotkeyA32.exe, AutoHotkeyU32.exe and (if present) AutoHotkeyU64.exe to AutoHotkey\*\_UIA.exe.
- Sets the `uiAccess` attribute in each UIA file's embedded manifest.
- Creates a self-signed digital certificate named "AutoHotkey" and signs each UIA file.
- Registers the context menu option to run the appropriate exe file.

If any these UIA files are present before installation, the installer will automatically update them even if the UI Access option is not enabled.

For command-line installations, specify `/uiAccess` or `/uiAccess=1` to

enable the option and `/uiAccess=0` to disable it. By default, the installer will enable the option if UAC is enabled and the UI Access context menu option was present before installation.

Scripts which need to run other scripts with UI access can simply [Run](#) the appropriate UIA.exe file with the normal [command line parameters](#).

### Known limitations:

- UIA is only effective if the file is in a trusted location; i.e. a Program Files sub-directory.
- UIA.exe files created on one computer cannot run on other computers without first installing the digital certificate which was used to sign them.
- UIA.exe files cannot be started via `CreateProcess` due to security restrictions. `ShellExecute` can be used instead. [Run](#) tries both.
- UIA.exe files cannot be modified, as it would invalidate the file's digital signature.
- Because UIA programs run at a different "integrity level" than other programs, they can only access objects registered by other UIA programs. For example, `ComObjActive("Word.Application")` will fail because Word is not marked for UI Access.
- The script's own windows can't be automated by non-UIA programs/scripts for security reasons.
- Running a non-UIA script which uses a mouse hook (even as simple as `#InstallMouseHook`) may prevent all mouse hotkeys from working when the mouse is pointing at a window owned by a UIA script, even

hotkeys implemented by the UIA script itself. A workaround is to ensure UIA scripts are loaded last.

For more details, see [Enable interaction with administrative programs](#) on the archive forum.

## Script Showcase

See [this page](#) for some useful scripts.

# Variables and Expressions

## Table of Contents

- Variables
- Expressions
- Operators in Expressions
- Built-in Variables
- Environment Variables vs. Normal Variables
- Variable Capacity and Memory

## Variables

**Variable types:** AutoHotkey has no explicitly defined variable types. Instead, any variable can contain a string, integer, floating-point number, or reference to an object. Additionally, numbers are automatically converted to or from strings as required. The `Type` function can be used to determine the actual type of a value.

**Variable scope and declarations:** With the exception of `local variables` in functions, all variables are global; that is, their contents may be read or altered by any part of the script. Except where noted on the `functions page`, variables do not need to be declared; they come into existence simply by using them (and each variable starts off empty/blank).

**Variable names:** Variable names are not case sensitive (for example, `CurrentDate` is the same as `currentdate`). Variable names may be up to 253 characters long and may consist of letters, numbers, underscore and non-ASCII characters. Variable names must not start with a digit.

**Reserved words:** `and`, `contains`, `in`, `is`, `new`, `not` and `or`. These words are reserved for use as `operators` and therefore cannot be used as variable names when written literally in an expression.

Names of control flow statements are also reserved, primarily to detect mistakes. This includes: `Break`, `Catch`, `Continue`, `Else`, `Finally`, `For`, `Gosub`, `Goto`, `If`, `Loop`, `LoopFile`, `LoopParse`, `LoopRead`, `LoopReg`,

Return, Throw, Try, Until, While

Functions, classes and window groups use the same validation as variables and therefore have the same restrictions.

**Storing values in variables:** To store a string or number in a variable, use the colon-equal operator (`:=`) followed by a number, quoted string or any other type of expression. For example:

```
MyNumber := 123
MyString := "This is a literal string."
CopyOfVar := Var
```

Variables which have not been assigned a value contain an empty string by default. Therefore, to erase the contents of a variable, simply assign an empty string:

```
MyVar := ""
```

A variable can also be assigned a value indirectly, by using it as an *output variable* of a command. For example:

```
MouseGetPos x, y
```

**Retrieving the contents of variables:** To include the contents of a variable in a string, use concatenation or `Format`. For example:

```
MsgBox "The value of Var is " . Var . "."
```

```
MsgBox "The value in the variable named Var is
" Var "."
MsgBox Format("Var has the value {1}.", Var)
```

Sub-expressions can be combined with strings in the same way. For example:

```
MsgBox("The sum of X and Y is " & (X + Y))
```

**Comparing variables:** Please read the expressions section below for important notes about the different kinds of comparisons.

## Expressions

Expressions are used to perform one or more operations upon a series of variables, literal strings, and/or literal numbers.

Variable names in an expression are not enclosed in percent signs (except for variables inside quoted strings, and [pseudo-arrays](#) and other [double references](#)). Consequently, literal strings must be enclosed in double quotes to distinguish them from variables. For example:

```
if (CurrentSetting > 100 or FoundColor <>
 "Blue")
 MsgBox "The setting is too high or the
 wrong color is present."
```

In the example above, "Blue" appears in quotes because it is a literal string. Single-quote marks (') and double-quote marks (") function identically, except that a string enclosed in single-quote marks can contain literal double-quote marks and vice versa. Therefore, to include an *actual* quote mark inside a literal string, [escape](#) the quote mark or enclose the string in the opposite type of quote mark. For example:

```
MsgBox "She said, \"An apple a day.\""
MsgBox 'She said, "An apple a day."'
```

**Empty strings:** To specify an empty string in an expression, use an empty pair of quotes. For example, the statement `if (MyVar <> "")` would be true if

*MyVar* is not blank.

**Storing the result of an expression:** To assign a result to a variable, use the `:=` operator. For example:

```
NetPrice := Price * (1 - Discount/100)
```

**Boolean values:** When an expression is required to evaluate to true or false (such as an IF-statement), a blank or zero result is considered false and all other results are considered true. For example, the statement `if ItemCount` would be false only if ItemCount is blank or 0. Similarly, the expression `if not ItemCount` would yield the opposite result.

Operators such as NOT/>/=/< automatically produce a true or false value: they yield 1 for true and 0 for false. However, the AND/OR operators always produce one of the input values. For example, in the following expression, the variable *Done* is assigned 1 if A\_Index is greater than 5 or the value of *FoundIt* in all other cases:

```
Done := A_Index > 5 or FoundIt
```

As hinted above, a variable can be used to hold a false value simply by making it blank or assigning 0 to it. To take advantage of this, the shorthand statement `if Done` can be used to check whether the variable *Done* is true or false.

In an expression, the keywords *true*, *false* and *null* resolve to 1, 0 and 0. They can be used to make a script more readable as in these examples:

```
CaseSensitive := false
ContinueSearch := true
hWindow := null
```

**Integers and floating point:** Within an expression, numbers are considered to be floating point if they contain a decimal point or scientific notation; otherwise, they are integers. For most operators -- such as addition and multiplication -- if either of the inputs is a floating point number, the result will also be a floating point number.

Within expressions and non-expressions alike, integers may be written in either hexadecimal or decimal format. Hexadecimal numbers all start with the prefix 0x. For example, `Sleep 0xFF` is equivalent to `Sleep 255`. Floating point numbers can optionally be written in scientific notation, with or without a decimal point (e.g. `1e4` or `-2.1E-4`).

Within expressions, unquoted literal numbers such as `128`, `0x7F` and `1.0` are converted to pure numbers before the script begins executing, so converting the number to a string may produce a value different to the original literal value. For example:

```
MsgBox(0x7F) ; Shows 128
MsgBox(1.00) ; Shows 1.0
```

## Operators in Expressions

Operators of equal precedence such as multiply (\*) and divide (/) are evaluated in left-to-right order unless otherwise specified below. By contrast, an operator of lower precedence such as add (+) is evaluated *after* a higher one such as multiply (\*). For example,  $3 + 2 * 2$  is evaluated as  $3 + (2 * 2)$ . Parentheses may be used to override precedence as in this example:  $(3 + 2) * 2$

Except where noted below, any blank value (empty string) or non-numeric value involved in a math operation is **not** assumed to be zero. Instead, it is treated as an error, which causes that part of the expression to evaluate to an empty string. For example, if the variable X is blank, the expression X+1 yields a blank value rather than 1.

## Expression Operators (in descending precedence order)

Evaluates the sub-expression *Expr* and uses its value as the name or partial name of a variable or function. This allows the script to refer to a variable or function whose name is not written literally in the script, but is determined by evaluating *Expr*, which is typically another variable. Percent signs cannot be used directly within *Expr* due to ambiguity, but can be nested within parentheses. Otherwise, *Expr* can be any expression.

`%Expr%( )` performs a [dynamic function call](#).

`%Expr%` dynamically retrieves a variable by name. The result of the sub-expression *Expr* must be the name or partial name of the variable to be retrieved. If there are any adjoining

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>%Expr%</b></p> | <p><i>%Expr%</i> sequences and partial <a href="#">variable names</a> (without any spaces or other characters between them), they are combined to form a single name.</p> <p>If the variable does not already exist, a blank variable is created. An <a href="#">exception</a> is thrown if the name is <a href="#">invalid</a>.</p> <p>This is most commonly used to reference <a href="#">pseudo-array</a> elements such as in the following example:</p> <pre data-bbox="509 613 1349 804"> Loop 4     MsgBox("IP address " A_Index " is " <a href="#">A_IPAddress</a><a href="#">%A_Index%</a>) </pre> <p>Although this is historically known as a "double-deref", this term is inaccurate when <i>Expr</i> does not contain a variable (first deref), and also when the resulting variable is the target of an assignment, not being dereferenced (second deref).</p> |
| <p><b>x.y</b></p>    | <p><b>Object access.</b> Get or set a value or call a method of object <i>x</i>, where <i>y</i> is a literal value. See <a href="#">object syntax</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <p><b>new</b></p>    | <p><code>new <a href="#">Class</a></code> or <code>new <a href="#">Class</a> (<a href="#">Params</a>)</code> creates a new object derived from <i>Class</i>. For example, <code>x := new y</code> is often equivalent to <code>x := {base: y}</code>. <i>Class</i> can be almost any expression which produces a class object, but is typically a class name such as <code><a href="#">GlobalClass</a></code> or <code><a href="#">GlobalClass.NestedClass</a></code>. More complex cases such as <code>new new (getClass()) (params1) (params2)</code> are also supported, but there must be no space between the class expression and parameter list. For details, see <a href="#">Custom Objects</a>.</p>                                                                                                                                                               |
|                      | <p><b>Pre- and post-increment/decrement.</b> Adds or subtracts 1 from a variable. The operator may appear either before or after the variable's name. If it appears <i>before</i> the name, the operation is performed immediately and its result is used by</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>++<br/>--</p> | <p>the next operation. For example, <code>Var := ++X</code> increments X immediately and then assigns its value to <i>Var</i>. Conversely, if the operator appears <i>after</i> the variable's name, the operation is performed <i>after</i> the variable is used by the next operation. For example, <code>Var := X++</code> increments X only after assigning the current value of X to <i>Var</i>. Due to backward compatibility, the operators ++ and -- treat blank variables as zero, but only when they are alone on a line; for example, <code>y:=1, ++X</code> and <code>MsgBox ++X</code> both produce a blank result when x is blank.</p> <p>If the target variable is empty, the operators ++ and -- treat it as zero. This allows a line such as <code>Var++</code> to be used in a hotkey without <i>Var</i> having been initialized in the <a href="#">auto-execute section</a>.</p> |
| <p>**</p>        | <p><b>Power.</b> Both the base and the exponent may contain a decimal point. If the exponent is negative, the result will be formatted as a floating point number even if the base and exponent are both integers. Since ** is of higher precedence than unary minus, <code>-2**2</code> is evaluated like <code>-(2**2)</code> and so yields -4. Thus, to raise a literal negative number to a power, enclose it in parentheses such as <code>(-2)**2</code>. Note: A negative base combined with a fractional exponent such as <code>(-2)**0.5</code> is not supported; it will yield an empty string. But both <code>(-2)**2</code> and <code>(-2)**2.0</code> are supported.</p>                                                                                                                                                                                                                |
|                  | <p><b>Unary minus (-):</b> Inverts the sign of its operand.</p> <p><b>Unary plus (+):</b> <code>+N</code> is equivalent to <code>-(-N)</code>. This has no effect when applied to a pure number, but can be used to convert numeric strings to pure numbers. As with other math operators, the result is a blank value (empty string) if the operand is not a number or numeric string.</p> <p><b>Logical-not (!):</b> If the operand is blank or 0, the result of applying logical-not is 1, which means "true". Otherwise, the result is 0 (false). For example: <code>!x or !(y and z)</code>. Note: The word NOT is synonymous with ! except that ! has a</p>                                                                                                                                                                                                                                   |

-  
!  
~  
& \*

higher precedence. Consecutive unary operators such as `!!Var` are allowed because they are evaluated in right-to-left order.

**Bitwise-not (~):** This inverts each bit of its operand. If the operand is a floating point value, it is truncated to an integer prior to the calculation. If the operand is between 0 and 4294967295 (0xffffffff), it will be treated as an unsigned 32-bit value. Otherwise, it is treated as a signed 64-bit value. For example, `~0xf0f` evaluates to 0xffff0f0 (4294963440).

**Address (&):** `&MyVar` retrieves the address of *MyVar*'s contents in memory, which is typically used with [DllCall structures](#). *MyVar*'s contents can be a string, a 64-bit integer, a 64-bit floating-point number, or an object.

\*  
/  
//

**Multiply (\*):** The result is an integer if both inputs are integers; otherwise, it is a floating point number.

**True divide (/):** True division yields a floating point result even when both inputs are integers. For example, `3/2` yields 1.5 rather than 1, and `4/2` yields 2.0 rather than 2.

**Floor divide (//):** The double-slash operator uses high-performance integer division if the two inputs are integers. For example, `5//3` is 1 and `5//-3` is -1. If either of the inputs is in floating point format, floating point division is performed and the result is truncated to the nearest integer to the left. For example, `5//3.0` is 1.0 and `5.0//-3` is -2.0. Although the result of this floating point division is an integer, it is stored in floating point format so that anything else that uses it will see it as such. For modulo, see [mod\(\)](#).

The `*=` and `/=` operators are a shorthand way to multiply or divide the value in a variable by another value. For example, `Var*=2` produces the same result as `Var:=Var*2` (though the former performs better).

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            | Division by zero yields a blank result (empty string).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| +<br>-     | <b>Add (+) and subtract (-).</b> On a related note, the += and -= operators are a shorthand way to increment or decrement a variable. For example, <code>Var+=2</code> produces the same result as <code>Var:=Var+2</code> (though the former performs better). Similarly, a variable can be increased or decreased by 1 by using <code>Var++</code> , <code>Var--</code> , <code>++Var</code> , or <code>--Var</code> .                                                                                                                                   |
| <<<br>>>   | <b>Bit shift left (&lt;&lt;) and right (&gt;&gt;).</b> Example usage: <code>Value1 &lt;&lt; Value2</code> . Any floating point input is truncated to an integer prior to the calculation. Shift left (<<) is equivalent to multiplying <i>Value1</i> by "2 to the <i>Value2</i> th power". Shift right (>>) is equivalent to dividing <i>Value1</i> by "2 to the <i>Value2</i> th power" and rounding the result to the nearest integer leftward on the number line; for example, <code>-3&gt;&gt;1</code> is -2.                                          |
| &<br>&<br> | <b>Bitwise-and (&amp;), bitwise-exclusive-or (^), and bitwise-or ( ).</b> Of the three, & has the highest precedence and   has the lowest. Any floating point input is truncated to an integer prior to the calculation.                                                                                                                                                                                                                                                                                                                                   |
| .          | <b>Concatenate.</b> The period (dot) operator is used to combine two items into a single string (there must be at least one space on each side of the period). You may also omit the period to achieve the same result (except where ambiguous such as <code>x -y</code> , or when the item on the right side has a leading ++ or --). When the dot is omitted, there must be at least one space between the items to be merged.<br><br><pre> Var := "The color is " . FoundColor ; Explicit concat Var := "The color is " FoundColor ; Auto-concat </pre> |

|                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  | <p>Sub-expressions can also be concatenated. For example: <code>Var := "The net price is " . Price * (1 - Discount/100)</code>.</p> <p>A line that begins with a period (or any other operator) is automatically <a href="#">appended</a> to the line above it.</p> <p>The entire <a href="#">length</a> of each input is used, even if it includes binary zero. For example, <code>Chr(0x2010) Chr(0x0000) Chr(0x4030)</code> produces the following string of bytes (due to UTF-16-LE encoding): 0x10, 0x20, 0, 0, 0x30, 0x40. The result has an additional null-terminator (binary zero) which is not included in the length.</p> |
| <p>~=</p>                        | <p>Shorthand for <a href="#">RegexMatch</a>. For example, the result of <code>"abc123" ~= "\d"</code> is 4 (the position of the first numeric character).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <p>&gt; &lt;<br/>&gt;= &lt;=</p> | <p><b>Greater (&gt;), less (&lt;), greater-or-equal (&gt;=), and less-or-equal (&lt;=).</b> If either of the inputs is not numeric (or both are strings), they are compared alphabetically. For example, <code>2 &lt; "10"</code> is <i>true</i> whereas <code>"2" &lt; "10"</code> is <i>false</i>. The comparison is case sensitive only if <a href="#">StringCaseSense</a> has been turned on. See also: <a href="#">Sort</a></p> <p>When comparing strings, these operators compare only up to the first binary zero.</p>                                                                                                        |
| <p>=<br/>==<br/>&lt;&gt; !=</p>  | <p><b>Equal (=), case-sensitive-equal (==), and not-equal (&lt;&gt; or !=).</b> The operators != and &lt;&gt; are identical in function. The == operator behaves identically to = except when either of the inputs is not numeric (or both are strings), in which case == is always case sensitive and = is always case insensitive (the method of insensitivity depends on <a href="#">StringCaseSense</a>). By contrast, &lt;&gt; and != obey <a href="#">StringCaseSense</a>.</p> <p>The == operator can be used to compare strings which contain</p>                                                                             |

|                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                      | <p>binary zero. All other comparison operators except <code>~=</code> compare only up to the first binary zero.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <p><b>IS<br/>IN<br/>CONTAINS</b></p> | <p><code> Value is Type </code>: Yields true (1) if <i>value</i> is of the given <i>type</i> or false (0) otherwise. For details, see <i>Value is Type</i>.</p> <p><code> in </code> and <code> contains </code> are reserved for future use.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><b>NOT</b></p>                    | <p><b>Logical-NOT</b>. Except for its lower precedence, this is the same as the <code>!</code> operator. For example, <code>not (x = 3 or y = 3)</code> is the same as <code>!(x = 3 or y = 3)</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <p><b>AND<br/>&amp;&amp;</b></p>     | <p>Both of these are <b>logical-AND</b>. For example: <code>x &gt; 3 and x &lt; 10</code>.</p> <p>In the expression <code>x and y</code>, if <i>x</i> is <b>true</b>, the result is <i>y</i>. Otherwise the result is <i>x</i>, and <i>y</i> is not evaluated at all - <b>short-circuit evaluation</b> is applied to enhance performance. Although the result is not limited to boolean 0 and 1, it can be interpreted as a <b>boolean</b> value as with any other expression. In effect, the result is true only if both inputs are true.</p> <p>A line that begins with <code>AND/OR/&amp;&amp;/  </code> (or any other operator) is automatically <b>appended</b> to the line above it.</p> |
| <p><b>OR<br/>  </b></p>              | <p>Both of these are <b>logical-OR</b>. For example: <code>x &lt;= 3 or x &gt;= 10</code>.</p> <p>In the expression <code>x or y</code>, if <i>x</i> is <b>false</b>, the result is <i>y</i>. Otherwise the result is <i>x</i>, and <i>y</i> is not evaluated at all - <b>short-circuit evaluation</b> is applied to enhance performance. Although the result is not limited to boolean 0 and 1, it can be interpreted as a <b>boolean</b> value as with any other expression. In effect, the result is true if either input is true.</p>                                                                                                                                                      |

|                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ?:                                                                      | <p><b>Ternary operator.</b> This operator is a shorthand replacement for the <a href="#">if-else statement</a>. It evaluates the condition on its left side to determine which of its two branches should become its final result. For example, <code>var := x&gt;y ? 2 : 3</code> stores 2 in <i>Var</i> if x is greater than y; otherwise it stores 3. To enhance performance, only the winning branch is evaluated (see <a href="#">short-circuit evaluation</a>).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| :=<br>+=<br>-=<br>*=<br>/=<br>//=<br>.=<br> =<br>&=<br>^=<br>>>=<br><<= | <p><b>Assign.</b> Performs an operation on the contents of a variable and stores the result back in the same variable. The simplest assignment operator is <a href="#">colon-equals (:=)</a>, which stores the result of an expression in a variable. For a description of what the other operators do, see their related entries in this table. For example, <code>var //= 2</code> performs <a href="#">floor division</a> to divide <i>Var</i> by 2, then stores the result back in <i>Var</i>. Similarly, <code>var .= "abc"</code> is a shorthand way of writing <code>var := var . "abc"</code>.</p> <p>Unlike most other operators, assignments are evaluated from right to left. Consequently, a line such as <code>var1 := var2 := 0</code> first assigns 0 to <i>Var2</i> then assigns <i>Var2</i> to <i>Var1</i>.</p> <p>If an assignment is used as the input for some other operator, its value is the variable itself. For example, the expression <code>(var+=2) &gt; 50</code> is true if the newly-increased value in <i>Var</i> is greater than 50. This also allows an assignment to be passed <a href="#">ByRef</a>, or its <a href="#">address</a> taken; for example: <code>&amp;(x:="abc")</code>.</p> <p>The precedence of the assignment operators is automatically raised when it would avoid a syntax error or provide more intuitive behavior. For example: <code>not x:=y</code> is evaluated as <code>not (x:=y)</code>. Similarly, <code>++var := x</code> is evaluated as <code>++(var := x)</code>; and <code>z&gt;0 ? x:=2 : y:=2</code> is evaluated as <code>z&gt;0 ? (x:=2) : (y:=2)</code>.</p> <p>If the target variable is empty, the operators += and -= treat it</p> |

|                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                  | as zero. This allows a line such as <code>Var += 1</code> to be used in a hotkey without <code>Var</code> having been initialized in the <a href="#">auto-execute section</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ,                                                                | <p><b>Comma (multi-statement).</b> Commas may be used to write multiple sub-expressions on a single line. This is most commonly used to group together multiple assignments or function calls. For example: <code>x:=1, y+=2, ++index, func()</code>. Such statements are executed in order from left to right. Note: A line that begins with a comma (or any other operator) is automatically <a href="#">appended to the line above it</a>. See also: <a href="#">comma performance</a>.</p> <p>When a multi-statement expression is used inside another expression, each sub-expression is evaluated and the value of the final (rightmost) sub-expression becomes the result of the expression. For example, <code>x := (++y, 1)</code> increments <code>y</code> and assigns the number 1 to <code>x</code>.</p> |
| <code>mod()</code><br><code>round()</code><br><code>abs()</code> | These and other built-in math functions are described <a href="#">here</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>%func%()</code>                                            | See <a href="#">Dynamically Calling a Function</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>F(p*)</code>                                               | See <a href="#">Variadic Functions</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>x[y]</code>                                                | <b>Object access.</b> Get or set a value or call a method of object <code>x</code> , where <code>y</code> is a parameter list or calculated method name. See <a href="#">array syntax</a> and <a href="#">object syntax</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

**Performance:** The comma operator is usually faster than writing separate

expressions, especially when assigning one variable to another (e.g. `x:=y`, `a:=b`). Performance continues to improve as more and more expressions are combined into a single expression; for example, it may be 35% faster to combine five or ten simple expressions into a single expression.

## Built-in Variables

The following variables are built into the program and can be referenced by any script. Except where noted, these variables are read-only; that is, their contents cannot be directly altered by the script.

### Table of Contents

- Special Characters: `A_Space`, `A_Tab`
- Script Properties: `command line parameters`, `A_WorkingDir`, `A_ScriptDir`, `A_ScriptName`, (...more...)
- Date and Time: `A_YYYY`, `A_MM`, `A_DD`, `A_Hour`, `A_Min`, `A_Sec`, (...more...)
- Script Settings: `A_IsSuspended`, `A_TitleMatchMode`, (...more...)
- User Idle Time: `A_TimeIdle`, `A_TimeIdlePhysical`
- Hotkeys, Hotstrings, and Custom Menu Items: `A_ThisHotkey`, `A_EndChar`, `A_ThisMenuItem`, (...more...)
- Operating System and User Info: `A_OSVersion`, `A_ScreenWidth`, `A_ScreenHeight`, (...more...)
- Misc: `A_Cursor`, `A_CaretX`, `A_CaretY`, `A_EventInfo`, `Clipboard`, `ClipboardAll`, `ErrorLevel`
- Loop: `A_Index`, (...more...)
- Other: `A_ScriptStruct`, `A_GlobalStruct`, `A_IsDll` (...more...)

### Special Characters

|         |                                    |
|---------|------------------------------------|
| A_Space | Contains a single space character. |
| A_Tab   | Contains a single tab character.   |

## Script Properties

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_Args              | <b>Read/write:</b> Contains an <a href="#">array</a> of command line parameters. For details, see <a href="#">Passing Command Line Parameters to a Script</a> .                                                                                                                                                                                                                                                                                                                           |
| A_WorkingDir        | <b>Read/write:</b> The script's current working directory, which is the directory where files will be accessed by default. The final backslash is not included unless it is the root directory. Two examples are <code>C:\</code> and <code>C:\My Documents</code> .<br><br>Use <a href="#">SetWorkingDir</a> or assign a path to A_WorkingDir to change the working directory.<br><br>The script's working directory defaults to A_ScriptDir, regardless of how the script was launched. |
| A_InitialWorkingDir | The script's initial working directory, which is determined by how it was launched. For example, if it was run via a shortcut such as on the Start Menu -- its initial working directory is determined by the "Start in" field within the shortcut's properties.                                                                                                                                                                                                                          |
| A_ScriptDir         | The full path of the directory where the current script is located. The final backslash is omitted (even for root directories).                                                                                                                                                                                                                                                                                                                                                           |
| A_ScriptName        | <b>Read/write:</b> The default title for MsgBox, InputBox, FileOpen, FileSelect and GuiCreate. If not set by the script, it defaults to the file name of the current script, without its path, e.g. MyScript.ahk.                                                                                                                                                                                                                                                                         |
| A_ScriptFullPath    | The full path of the current script, e.g. C:\My Documents\MyScript.ahk                                                                                                                                                                                                                                                                                                                                                                                                                    |
| A_ScriptHwnd        | The unique ID (HWND/handle) of the script's hidden main window.                                                                                                                                                                                                                                                                                                                                                                                                                           |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_LineNumber | <p>The number of the currently executing line within the script (or one of its <a href="#">#Include files</a>). This line number will match the line number shown by <a href="#">ListLines</a>; it can be useful for error reporting such as in this example: <code>MsgBox "Could not write to 1 file (line number " A_LineNumber ")".</code></p> <p>Since a <a href="#">compiled script</a> has merged all its <a href="#">#Include files</a> into one big script, its line numbering may be different than when run in non-compiled mode.</p>                                                                                                                |
| A_LineFile   | <p>The full path and name of the file to which <a href="#">A_LineNumber</a> belongs, which will be the same as <a href="#">A_ScriptFullPath</a> unless the line belongs to one of a non-compiled script's <a href="#">#Include files</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                  |
| A_ThisFunc   | <p>The name of the <a href="#">user-defined function</a> that is currently executing (blank if none); for example: MyFunction. See <a href="#">IsFunc()</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| A_ThisLabel  | <p>The name of the <a href="#">label (subroutine)</a> that is currently executing (blank if none); for example: MyLabel. It is updated when the script executes <a href="#">Gosub/Return</a> or <a href="#">Goto</a>. It is also updated for automatically-called labels such as <a href="#">timers</a>, <a href="#">GUI threads</a>, <a href="#">items</a>, <a href="#">hotkeys</a> and <a href="#">hotstrings</a>. However, <a href="#">A_ThisLabel</a> is updated when execution "falls into" a label from above; that happens, <a href="#">A_ThisLabel</a> retains its previous value. See <a href="#">A_ThisHotkey</a> and <a href="#">IsLabel()</a>.</p> |
| A_AhkVersion | <p>Contains the version of AutoHotkey that is running the script, such as 1.0.22. In the case of a <a href="#">compiled script</a>, the version that was originally used to compile it is reported. The format of the version number allows a script to check whether <a href="#">A_AhkVersion</a> is greater than some minimum version number with <code>&gt;</code> or <code>&gt;=</code> as in this example: <code>If (A_AhkVersion &lt; "1.0.25.07").</code></p>                                                                                                                                                                                           |
|              | <p>For non-compiled scripts: The full path and name of the file that is actually running the current script. For example: <code>C:\Program Files\AutoHotkey\AutoHotkey.exe</code></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

|              |                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_AhkPath    | For <b>compiled scripts</b> : The same as the above except the AutoHotkey directory is discovered via the registry entry <i>HKEY_LOCAL_MACHINE\SOFTWARE\AutoHotkey\In</i> . If there is no such entry, A_AhkPath is blank.                                                                                                                                                                            |
| A_DllPath    | The full path and name of the DLL file that is actually running the current script. For example: C:\Program Files\AutoHotkey\AutoHotkey.dll. If the script is executed by EXE, it contains the path of EXE                                                                                                                                                                                            |
| A_AhkDir     | For non-compiled scripts: The full path and name of the directory of EXE file that is actually running the current script. For example: C:\Program Files\AutoHotkey<br><br>For <b>compiled scripts</b> : The same as the above except the AutoHotkey directory is discovered via the registry entry <i>HKEY_LOCAL_MACHINE\SOFTWARE\AutoHotkey\In</i> . If there is no such entry, A_AhkPath is blank. |
| A_AhkDir     | The full path and name of the directory of DLL file that is actually running the current script. For example: C:\Program Files\AutoHotkey, if the script is executed by EXE it contains the directory of EXE.                                                                                                                                                                                         |
| A_IsCompiled | Contains 1 if the script is running as a <b>compiled EXE</b> and an empty string (which is considered <b>false</b> ) if it is not.                                                                                                                                                                                                                                                                    |

## Date and Time

|        |                                                                                                                                                                                           |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_YYYY | Current 4-digit year (e.g. 2004). Synonymous with A_Year. Note: To retrieve a formatted time or date appropriate for your locale and language, use <a href="#">FormatTime()</a> (time and |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|         |                                                                                                                                                                                                                                                                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         | long date) or <code>FormatTime(, "LongDate")</code> (retrieves long-format date).                                                                                                                                                                                                                                                                   |
| A_MM    | Current 2-digit month (01-12). Synonymous with A_Mon.                                                                                                                                                                                                                                                                                               |
| A_DD    | Current 2-digit day of the month (01-31). Synonymous with A_MDay.                                                                                                                                                                                                                                                                                   |
| A_MMMM  | Current month's full name in the current user's language, e.g. July                                                                                                                                                                                                                                                                                 |
| A_MMM   | Current month's abbreviation in the current user's language, e.g. Jul                                                                                                                                                                                                                                                                               |
| A_DDDD  | Current day of the week's full name in the current user's language, e.g. Sunday                                                                                                                                                                                                                                                                     |
| A_DDD   | Current day of the week's 3-letter abbreviation in the current user's language, e.g. Sun                                                                                                                                                                                                                                                            |
| A_WDay  | Current 1-digit day of the week (1-7). 1 is Sunday in all locales.                                                                                                                                                                                                                                                                                  |
| A_YDay  | Current day of the year (1-366). The value is not zero-padded, e.g. 9 is retrieved, not 009. To retrieve a zero-padded value, use the following: <code>FormatTime(, "YDay0")</code> .                                                                                                                                                               |
| A_YWeek | Current year and week number (e.g. 200453) according to ISO 8601. To separate the year from the week, use <code>SubStr</code> . Precise definition of A_YWeek: If the week containing January 1st has four or more days in the new year, it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. |
| A_Hour  | Current 2-digit hour (00-23) in 24-hour time (for example, 17 is 5pm). To retrieve 12-hour time as well as an AM/PM indicator, follow this example: <code>FormatTime(, "h:mm:ss tt")</code>                                                                                                                                                         |
| A_Min   | Current 2-digit minute (00-59).                                                                                                                                                                                                                                                                                                                     |
| A_Sec   | Current 2-digit second (00-59).                                                                                                                                                                                                                                                                                                                     |

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_MSec      | Current 3-digit millisecond (000-999). To remove the leading zeros, follow this example: <code>Milliseconds := A_MSec + 0</code> .                                                                                                                                                                                                                                                                                                                                                                              |
| A_Now       | The current local time in <code>YYYYMMDDHH24MISS</code> format. Note: Date and time math can be performed with <code>DateAdd</code> and <code>DateDiff</code> . Also, <code>FormatTime</code> can format the date and/or time according to your locale or preferences.                                                                                                                                                                                                                                          |
| A_NowUTC    | The current Coordinated Universal Time (UTC) in <code>YYYYMMDDHH24MISS</code> format. UTC is essentially the same as Greenwich Mean Time (GMT).                                                                                                                                                                                                                                                                                                                                                                 |
| A_TickCount | <p>The number of milliseconds since the computer was rebooted. By storing <code>A_TickCount</code> in a variable, elapsed time can later be measured by subtracting that variable from the latest <code>A_TickCount</code> value. For example:</p> <pre> StartTime := A_TickCount Sleep 1000 ElapsedTime := A_TickCount - StartTime MsgBox ElapsedTime " milliseconds have elapsed." </pre> <p>If you need more precision than <code>A_TickCount</code>'s 10ms, use <code>QueryPerformanceCounter()</code>.</p> |

## Script Settings

|               |                                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------|
| A_IsSuspended | Contains 1 if the script is <code>suspended</code> and 0 otherwise.                                                               |
| A_IsPaused    | Contains 1 if the <code>thread</code> immediately underneath the current thread is <code>paused</code> . Otherwise it contains 0. |

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_IsCritical                        | Contains 0 if <a href="#">Critical</a> is off for the <a href="#">current thread</a> . Otherwise it contains an integer greater than zero, namely the <a href="#">message-check frequency</a> being used by <a href="#">Critical</a> . The current state of <a href="#">Critical</a> can be saved and restored via <code>Old_IsCritical := A_IsCritical</code> followed later by <code>A_IsCritical := Old_IsCritical</code> . |
| A_TitleMatchMode                    | <b>Read/write:</b> The current mode set by <a href="#">SetTitleMatchMode</a> : 1, 2, 3, or <a href="#">RegEx</a> .                                                                                                                                                                                                                                                                                                             |
| A_TitleMatchModeSpeed               | <b>Read/write:</b> The current match speed (fast or slow) set by <a href="#">SetTitleMatchMode</a> .                                                                                                                                                                                                                                                                                                                           |
| A_DetectHiddenWindows               | <b>Read/write:</b> The current mode (On or Off) set by <a href="#">DetectHiddenWindows</a> .                                                                                                                                                                                                                                                                                                                                   |
| A_DetectHiddenText                  | <b>Read/write:</b> The current mode (On or Off) set by <a href="#">DetectHiddenText</a> .                                                                                                                                                                                                                                                                                                                                      |
| A_StringCaseSense                   | <b>Read/write:</b> The current mode (On, Off, or Locale) set by <a href="#">StringCaseSense</a> .                                                                                                                                                                                                                                                                                                                              |
| A_FileEncoding                      | <b>Read/write:</b> The default encoding for various commands; see <a href="#">FileEncoding</a> .                                                                                                                                                                                                                                                                                                                               |
| A_SendMode                          | <b>Read/write:</b> The current mode (Event, Input, Play or InputThenPlay) set by <a href="#">SendMode</a> .                                                                                                                                                                                                                                                                                                                    |
| A_SendLevel                         | <b>Read/write:</b> The current <a href="#">SendLevel</a> setting (an integer between 0 and 100, inclusive).                                                                                                                                                                                                                                                                                                                    |
| A_StoreCapslockMode                 | <b>Read/write:</b> The current mode (On or Off) set by <a href="#">SetStoreCapslockMode</a> .                                                                                                                                                                                                                                                                                                                                  |
| A_KeyDelay<br>A_KeyDuration         | <b>Read/write:</b> The current delay or duration set by <a href="#">SetKeyDelay</a> (always decimal, not hex).                                                                                                                                                                                                                                                                                                                 |
| A_KeyDelayPlay<br>A_KeyDurationPlay | <b>Read/write:</b> The current delay or duration set by <a href="#">SetKeyDelay</a> for the <a href="#">SendPlay</a> mode (always decimal, not hex).                                                                                                                                                                                                                                                                           |
|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                |

|                                                                                                   |                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_WinDelay                                                                                        | <b>Read/write:</b> The current delay set by <code>SetWinDelay</code> (always decimal, not hex).                                                                                                                                                                                                                                                |
| A_ControlDelay                                                                                    | <b>Read/write:</b> The current delay set by <code>SetControlDelay</code> .                                                                                                                                                                                                                                                                     |
| A_MouseDelay<br>A_MouseDelayPlay                                                                  | <b>Read/write:</b> The current delay set by <code>SetMouseDelay</code> (always decimal, not hex). <code>A_MouseDelay</code> is for the traditional <code>SendEvent</code> mode, whereas <code>A_MouseDelayPlay</code> is for <code>SendPlay</code> .                                                                                           |
| A_DefaultMouseSpeed                                                                               | <b>Read/write:</b> The current speed set by <code>SetDefaultMouseSpeed</code> .                                                                                                                                                                                                                                                                |
| A_CoordModeToolTip<br>A_CoordModePixel<br>A_CoordModeMouse<br>A_CoordModeCaret<br>A_CoordModeMenu | <b>Read/write:</b> The current mode (Client = 0, Window/Relative 0 1 or Screen = 2) set by <code>CoordMode</code> .                                                                                                                                                                                                                            |
| A_RegView                                                                                         | <b>Read/write:</b> The current registry view as set by <code>SetRegView</code> .                                                                                                                                                                                                                                                               |
| A_IconHidden                                                                                      | Contains 1 if the tray icon is currently hidden or 0 otherwise. The icon can be hidden via <code>#NoTrayIcon</code> or the <code>Menu</code> command.                                                                                                                                                                                          |
| A_IconTip                                                                                         | Blank unless a custom tooltip for the tray icon has been specified via <code>Menu "Tray", "Tip"</code> -- in which case it's the text of the tip.                                                                                                                                                                                              |
| A_IconFile                                                                                        | Blank unless a custom tray icon has been specified via <code>Menu "tray", "icon"</code> -- in which case it's the full path and name of the icon's file. <b>Known limitation:</b> This path may be incorrect if the script originally passed a relative path to a system DLL; for example, <code>Menu "Tray", "Icon", "user32.dll", 2</code> . |
| A_IconNumber                                                                                      | Blank if <code>A_IconFile</code> is blank. Otherwise, it's the number of the icon in <code>A_IconFile</code> (typically 1).                                                                                                                                                                                                                    |

## User Idle Time

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_TimeIdle         | <p>The number of milliseconds that have elapsed since the system last received keyboard, mouse, or other input. This is useful for determining whether the user is away. Physical input from the user as well as artificial input generated by <b>any</b> program or script (such as the <a href="#">Send</a> or <a href="#">MouseMove</a> commands) will reset this value back to zero. Since this value tends to increase by increments of 10, do not check whether it is equal to another value. Instead, check whether it is greater or less than another value. For example:</p> <pre data-bbox="626 793 1351 1033">if A_TimeIdle &gt; 600000     MsgBox "The last keyboard or mouse activity was at least 10 minutes ago."</pre> |
| A_TimeIdlePhysical | <p>Similar to above but ignores artificial keystrokes and/or mouse clicks whenever the corresponding hook (<a href="#">keyboard</a> or <a href="#">mouse</a>) is installed; that is, it responds only to physical events. (This prevents simulated keystrokes and mouse clicks from falsely indicating that a user is present.) If neither hook is installed, this variable is equivalent to A_TimeIdle. If only one hook is installed, only its type of physical input affects A_TimeIdlePhysical (the other/non-installed hook's input, both physical and artificial, has no effect).</p>                                                                                                                                            |

## Hotkeys, Hotstrings, and Custom Menu Items

|                |                                                                                          |
|----------------|------------------------------------------------------------------------------------------|
| A_ThisMenuItem | The name of the most recently selected <a href="#">custom menu item</a> (blank if none). |
|----------------|------------------------------------------------------------------------------------------|

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_ThisMenu        | The name of the menu from which A_ThisMenuItem was selected.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| A_ThisMenuItemPos | A number indicating the <u>current</u> position of A_ThisMenuItem within A_ThisMenu. The first item in the menu is 1, the second is 2, and so on. Menu separator lines are counted. This variable is blank if A_ThisMenuItem is blank or no longer exists within A_ThisMenu. It is also blank if A_ThisMenu itself no longer exists.                                                                                                                                                                                                                                                                                                      |
| A_ThisHotkey      | <p>The most recently executed <u>hotkey</u> or <u>non-auto-replace hotstring</u> (blank if none), e.g. #z. This value will change if the <u>current thread</u> is interrupted by another hotkey, so be sure to copy it into another variable immediately if you need the original value for later use in a subroutine.</p> <p>When a hotkey is first created -- either by the <u>Hotkey command</u> or a <u>double-colon label</u> in the script -- its key name and the ordering of its modifier symbols becomes the permanent name of that hotkey, shared by all <u>variants</u> of the hotkey.</p> <p>See also: <u>A_ThisLabel</u></p> |
| A_PriorHotkey     | Same as above except for the previous hotkey. It will be blank if none.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| A_PriorKey        | The name of the last key which was pressed prior to the most recent key-press or key-release, or blank if no applicable key-press can be found in the key history. All input generated by AutoHotkey scripts is excluded. For this variable to be of use, the <u>keyboard</u> or <u>mouse hook</u> must be installed and <u>key history</u> must be enabled.                                                                                                                                                                                                                                                                              |
|                   | The number of milliseconds that have elapsed                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|                        |                                                                                                                                                                                                                                   |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_TimeSinceThisHotkey  | since A_ThisHotkey was pressed. It will be -1 whenever A_ThisHotkey is blank.                                                                                                                                                     |
| A_TimeSincePriorHotkey | The number of milliseconds that have elapsed since A_PriorHotkey was pressed. It will be -1 whenever A_PriorHotkey is blank.                                                                                                      |
| A_EndChar              | The <a href="#">ending character</a> that was pressed by the user to trigger the most recent <a href="#">non-auto-replace hotstring</a> . If no ending character was required (due to the * option), this variable will be blank. |

## Operating System and User Info

|             |                                                                                                                                                                                                                                                                                                                                 |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_ComSpec   | Contains the same string as the environment's ComSpec variable (e.g. C:\Windows\system32\cmd.exe). Often used with <a href="#">Run/RunWait</a> .                                                                                                                                                                                |
| A_Temp      | The full path and name of the folder designated to hold temporary files (e.g. C:\DOCUME~1\UserName\LOCALS~1\Temp). It is retrieved from one of the following locations (in order): 1) the <a href="#">environment variables</a> TMP, TEMP, or USERPROFILE; 2) the Windows directory.                                            |
| A_OSVersion | <p>The version number of the operating system, in the format "<i>major.minor.build</i>". For example, Windows 7 SP1 is 6.1.7601.</p> <p>Applying compatibility settings in the AutoHotkey executable or compiled script's properties causes the OS to report a different version number, which is reflected by A_OSVersion.</p> |
| A_Is64bitOS | Contains 1 (true) if the OS is 64-bit or 0 (false) if it is 32-bit.                                                                                                                                                                                                                                                             |
|             |                                                                                                                                                                                                                                                                                                                                 |

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_PtrSize       | Contains the size of a pointer, in bytes. This is either 4 (32-bit) or 8 (64-bit), depending on what type of executable (EXE) is running the script.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| A_Language      | The system's default language, which is one of <a href="#">these 4-digit codes</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| A_ComputerName  | The name of the computer as seen on the network.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| A_UserName      | The logon name of the user who launched this script.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| A_WinDir        | The Windows directory. For example: C:\Windows                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| A_ProgramFiles  | <p>The Program Files directory (e.g. <code>C:\Program Files</code> or <code>C:\Program Files (x86)</code>). This is usually the same as the <code>ProgramFiles</code> <a href="#">environment variable</a>.</p> <p>On <a href="#">64-bit systems</a> (and not 32-bit systems), the following applies:</p> <ul style="list-style-type: none"> <li>• If the executable (EXE) that is running the script is 32-bit, A_ProgramFiles returns the path of the "Program Files (x86)" directory.</li> <li>• For 32-bit processes, the <code>ProgramW6432</code> <a href="#">environment variable</a> contains the path of the 64-bit Program Files directory. On Windows 7 and later, it is also set for 64-bit processes.</li> <li>• The <code>ProgramFiles(x86)</code> <a href="#">environment variable</a> contains the path of the 32-bit Program Files directory.</li> </ul> |
| A_AppData       | The full path and name of the folder containing the current user's application-specific data. For example: C:\Documents and Settings\Username\Application Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| A_AppDataCommon | The full path and name of the folder containing the all-users application-specific data.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_Desktop         | The full path and name of the folder containing the current user's desktop files.                                                                                                                                                                                                                                                                                                                            |
| A_DesktopCommon   | The full path and name of the folder containing the all-users desktop files.                                                                                                                                                                                                                                                                                                                                 |
| A_StartMenu       | The full path and name of the current user's Start Menu folder.                                                                                                                                                                                                                                                                                                                                              |
| A_StartMenuCommon | The full path and name of the all-users Start Menu folder.                                                                                                                                                                                                                                                                                                                                                   |
| A_Programs        | The full path and name of the Programs folder in the current user's Start Menu.                                                                                                                                                                                                                                                                                                                              |
| A_ProgramsCommon  | The full path and name of the Programs folder in the all-users Start Menu.                                                                                                                                                                                                                                                                                                                                   |
| A_Startup         | The full path and name of the Startup folder in the current user's Start Menu.                                                                                                                                                                                                                                                                                                                               |
| A_StartupCommon   | The full path and name of the Startup folder in the all-users Start Menu.                                                                                                                                                                                                                                                                                                                                    |
| A_MyDocuments     | The full path and name of the current user's "My Documents" folder. Unlike most of the similar variables, if the folder is the root of a drive, the final backslash is not included. For example, it would contain M: rather than M:\                                                                                                                                                                        |
| A_IsAdmin         | <p>If the current user has admin rights, this variable contains 1. Otherwise, it contains 0.</p> <p>To have the script restart itself as admin (or show a prompt to the user requesting admin), use <a href="#">Run *RunAs</a>. However, note that running the script as admin causes all programs launched by the script to also run as admin. For a possible alternative, see <a href="#">the FAQ</a>.</p> |
|                   |                                                                                                                                                                                                                                                                                                                                                                                                              |

|                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>A_ScreenWidth<br/>A_ScreenHeight</p> | <p>The width and height of the primary monitor, in pixels (e.g. 1024 and 768).</p> <p>To discover the dimensions of other monitors in a multi-monitor system, use <a href="#">SysGet</a>.</p> <p>To instead discover the width and height of the entire desktop (even if it spans multiple monitors), use the following example:</p> <pre style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">VirtualWidth := SysGet(78) VirtualHeight := SysGet(79)</pre> <p>In addition, use <a href="#">SysGet</a> to discover the work area of a monitor, which can be smaller than the monitor's total area because the taskbar and other registered desktop toolbars are excluded.</p> |
| <p>A_ScreenDPI</p>                      | <p>Number of pixels per logical inch along the screen width. In a system with multiple display monitors, this value is the same for all monitors. On most systems this is 96; it depends on the system's text size (DPI) setting. See also the GUI's <a href="#">-DPIScale</a> option.</p>                                                                                                                                                                                                                                                                                                                                                                                                         |
| <p>A_IPAddress1 through<br/>4</p>       | <p>The IP addresses of the first 4 network adapters in the computer.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## Misc.

```
SetTimer WatchCaret, 100
WatchCaret() {
 ToolTip, "X" A_CaretX " Y" A_CaretY,
A_CaretX, A_CaretY - 20
}
```

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_Cursor                      | <p>The type of mouse cursor currently being displayed. It will be one of the following words: AppStarting, Arrow, Cross, Help, IBeam, Icon, No, Size, SizeAll, SizeNESW, SizeNS, SizeNWSE, SizeWE, UpArrow, Wait, Unknown. The acronyms used with the size-type cursors are compass directions, e.g. NESW = NorthEast+SouthWest. The hand-shaped cursors (pointing and grabbing) are classified as Unknown.</p>                                                                                                                                                                     |
| A_CaretX<br>A_CaretY          | <p>The current X and Y coordinates of the caret (text insertion point). The coordinates are relative to the active window unless <a href="#">CoordMode</a> is used to make them relative to the entire screen. If there is no active window or the caret position cannot be determined, these variables are blank.</p> <p>The following script allows you to move the caret around to see its current position displayed in an auto-update tooltip. Note that some windows (e.g. certain versions of MS Word) report the same caret position regardless of its actual position.</p> |
| A_Input                       | <p>Current text of Input, used in multi-threading environment to retrieve or change the text of Input. It is also available after Input exited or before it started (see <a href="#">A</a> option).</p>                                                                                                                                                                                                                                                                                                                                                                             |
| A_GlobalStruct<br>(low level) | <p>Pointer to internal global structure.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| A_MainThreadID                | <p>Id of main thread (exe).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_ModuleHandle                | Handle of currently executed Module (dll/exe).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| A_ScriptStruct<br>(low level) | Pointer to internal script structure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| A_ThreadID                    | Handle of currently executed Thread.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| A_EventInfo                   | <p>Contains additional information about the following events:</p> <ul style="list-style-type: none"> <li>• <a href="#">Mouse wheel hotkeys</a> (WheelDown/Up/Left/Right)</li> <li>• <a href="#">RegisterCallback()</a></li> </ul> <p>Note: Unlike variables such as <a href="#">A_ThisHotkey</a>, each <a href="#">thread</a> retains its own value for <a href="#">A_EventInfo</a>. Therefore, if a thread is interrupted by another, upon being resumed it will still see its original/correct values in these variables.</p> <p><b>Read/write:</b> <a href="#">A_EventInfo</a> can also be set by the script, but can only accept unsigned integers within the range available to pointers (32-bit or 64-bit depending on the version of AutoHotkey).</p> |
| Clipboard                     | <b>Read/write:</b> The contents of the OS's clipboard, which can be read or written to. See the <a href="#">Clipboard</a> section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| ClipboardAll                  | <b>Read-only:</b> The entire contents of the clipboard (such as formatting and text). See <a href="#">ClipboardAll</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| ErrorLevel                    | <b>Read/write:</b> See <a href="#">ErrorLevel</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| A_LastError                   | <b>Read/write:</b> This is usually the result from the OS's <a href="#">GetLastError()</a> function after the script calls certain commands/functions, or the HRESULT of the last COM object invocation. See <a href="#">DllCall()</a> or <a href="#">Run/RunWait</a> for                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|  |                                                                                                                  |
|--|------------------------------------------------------------------------------------------------------------------|
|  | more details.<br><br>Assigning a value to A_LastError also causes the OS's SetLastError() function to be called. |
|--|------------------------------------------------------------------------------------------------------------------|

## Loop

|                      |                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_Index              | <b>Read/write:</b> This is the number of the current loop iteration (a 64-bit integer). For example, the first time the script executes the body of a loop, this variable will contain the number 1. For details see <a href="#">Loop</a> or <a href="#">While-loop</a> . |
| A_LoopFileName, etc. | This and other related variables are valid only inside a <a href="#">file-loop</a> .                                                                                                                                                                                      |
| A_LoopRegName, etc.  | This and other related variables are valid only inside a <a href="#">registry-loop</a> .                                                                                                                                                                                  |
| A_LoopReadLine       | See <a href="#">file-reading loop</a> .                                                                                                                                                                                                                                   |
| A_LoopField          | See <a href="#">parsing loop</a> .                                                                                                                                                                                                                                        |

## Environment Variables vs. "Normal" Variables

Environment variables are maintained by the operating system. You can see a list of them at the command prompt by typing SET then pressing Enter. Use [EnvGet](#) or [EnvSet](#) to retrieve or set environment variable.

A script may create a new environment variable or change the contents of an existing one with [EnvSet](#). However, such additions and changes are private; they are not seen by the rest of the system. One exception is when a script uses [Run](#) or [RunWait](#) to launch a program (even another script): such programs inherit a copy of the parent script's environment variables, including private ones.

## Variable Capacity and Memory

- When a variable is given a new string longer than its current contents, additional system memory is allocated automatically.
- The memory occupied by a large variable can be freed by setting it equal to nothing, e.g. `var := ""`.
- There is no limit to how many variables a script may create. The program is designed to support at least several million variables without a significant drop in performance.
- Commands, functions, and expressions that accept numeric inputs generally support 15 digits of precision for floating point values. For integers, 64-bit signed values are supported, which range from -9223372036854775808 (-0x8000000000000000) to 9223372036854775807 (0x7FFFFFFFFFFFFFFF). Any integer constants outside this range are not supported and might yield inconsistent results. By contrast, arithmetic operations on integers wrap around upon overflow (e.g.  $0x7FFFFFFFFFFFFFFF + 1 = -0x8000000000000000$ ).

# Functions

## Table of Contents

- [Introduction and Simple Examples](#)
- [Parameters](#)
- [Optional Parameters](#)
- [Returning Values to Caller](#)
- [Variadic Functions](#)
- [Local Variables](#)
- [Dynamically Calling a Function](#)
- [Short-circuit Boolean Evaluation](#)
- [Using Subroutines Within a Function](#)
- [Return, Exit, and General Remarks](#)
- [Using #Include to Share Functions Among Multiple Scripts](#)
- [Libraries of Functions: Standard Library and User Library](#)
- [Built-in Functions](#)
- [Macro Functions](#)

## Introduction and Simple Examples

A function is similar to a subroutine ([Gosub](#)) except that it can accept parameters (inputs) from its caller. In addition, a function may optionally return a value to its caller. Consider the following simple function that accepts two numbers and returns their sum:

```
Add(x, y)
{
 return x + y
}
```

The above is known as a function *definition* because it creates a function named "Add" (not case sensitive) and establishes that anyone who calls it must provide exactly two parameters (x and y). To call the function, assign its result to a variable with the `:=` operator. For example:

```
Var := Add(2, 3) ; The number 5 will be stored
in Var.
```

Also, a function may be called without storing its return value:

```
Add(2, 3)
Add 2, 3 ; Parentheses can be omitted if used
at the start of a line.
```

But in this case, any value returned by the function is discarded; so unless the function produces some effect other than its return value, the call would serve no

purpose.

Within an expression, a function call "evaluates to" the return value of the function. The return value can be assigned to a variable as shown above, or it can be used directly as shown below:

```
if InStr(MyVar, "fox")
 MsgBox "The variable MyVar contains the
word fox."
```

## Parameters

When a function is defined, its parameters are listed in parentheses next to its name (there must be no spaces between its name and the open-parenthesis). If a function does not accept any parameters, leave the parentheses empty; for example: `GetCurrentTimestamp()`.

**ByRef Parameters:** From the function's point of view, parameters are essentially the same as [local variables](#) unless they are defined as *ByRef* as in this example:

```
Swap(ByRef Left, ByRef Right)
{
 temp := Left
 Left := Right
 Right := temp
}
```

In the example above, the use of *ByRef* causes each parameter to become an alias for the variable passed in from the caller. In other words, the parameter and the caller's variable both refer to the same contents in memory. This allows the `Swap` function to alter the caller's variables by moving *Left*'s contents into *Right* and vice versa.

By contrast, if *ByRef* were not used in the example above, *Left* and *Right* would be copies of the caller's variables and thus the `Swap` function would have no external effect.

Since `return` can send back only one value to a function's caller, *ByRef* can be used to send back extra results. This is achieved by having the caller pass in a variable (usually empty) in which the function stores a value.

When passing large strings to a function, *ByRef* enhances performance and conserves memory by avoiding the need to make a copy of the string. Similarly, using *ByRef* to send a long string back to the caller usually performs better than something like `Return HugeString`.

If something other than a modifiable variable is passed to a *ByRef* parameter, the function behaves as though the keyword "ByRef" is absent. For example, `Swap(A_Index, i)` stores the value of *A\_Index* in *i*, but the value assigned to *Left* is discarded once the *Swap* function returns.

The `IsByRef()` function can be used to determine whether the caller supplied a variable for a given *ByRef* parameter.

Known limitations:

- It is not possible to pass properties of objects (such as `foo.bar`), `Clipboard` or other *built-in variables* to a function by reference. Instead, the function acts as though *ByRef* was omitted.
- Although a function may call itself recursively, if it passes one of its own *local variables* or non-*ByRef* parameters to itself *ByRef*, the new layer's *ByRef* parameter will refer to its own local variable of that name rather than the previous layer's. However, this issue does not occur when a function passes to itself a *global variable*, *static variable*, or *ByRef* parameter.

- If a parameter in a function-call resolves to a variable (e.g. `Var` or `++Var` or `var*=2`), other parameters to its left or right can alter that variable before it is passed to the function. For example, `func(Var, Var++)` would unexpectedly pass 1 and 0 when `Var` is initially 0, even when the function's first parameter is not *ByRef*. Since this behavior is counterintuitive, it might change in a future release.
- *ByRef* is not directly supported in functions called by COM clients, or when calling COM methods. Instead, the script receives or must pass a [wrapper object](#) containing the *VarType* and address of the value.

## Optional Parameters

When defining a function, one or more of its parameters can be marked as optional. This is done by appending the assignment operator followed by a default value. The following function has its Z parameter marked optional:

```
Add(X, Y, Z := 0) {
 return X + Y + Z
}
```

When the caller passes **three** parameters to the function above, Z's default value is ignored. But when the caller passes only **two** parameters, Z automatically receives the value 0.

It is not possible to have optional parameters isolated in the middle of the parameter list. In other words, all parameters that lie to the right of the first optional parameter must also be marked optional. However, optional parameters may be omitted from the middle of the parameter list when calling the function, as shown below:

```
Func(1,, 3)
Func(X, Y:=2, Z:=0) { ; Note that Z must still
 be optional in this case.
 MsgBox X ", " Y ", " Z
}
```

**ByRef parameters** also support default values; for example: `Func(ByRef p1 := "")`. Whenever the caller omits such a parameter, the function creates a

local variable to contain the default value; in other words, the function behaves as though the keyword "ByRef" is absent.

A parameter's default value must be one of the following: `true`, `false`, a literal integer, a literal floating point number, or a quoted/literal string such as "fox" or "".

## Returning Values to Caller

As described in [introduction](#), a function may optionally [return](#) a value to its caller.

```
MsgBox returnTest()

returnTest() {
 return 123
}
```

If you want to return extra results from a function, you may also use [ByRef](#):

```
returnByRef(A,B,C)
MsgBox A "," B "," C

returnByRef(ByRef val1, ByRef val2, ByRef val3)
{
 val1 := "A"
 val2 := 100
 val3 := 1.1
 return
}
```

Objects and Arrays can be used to return multiple values or even named values:

```
Test1 := returnArray1()
MsgBox Test1[1] "," Test1[2]

Test2 := returnArray2()
MsgBox Test2[1] "," Test2[2]
```



## Variadic Functions

When defining a function, write an asterisk after the final parameter to mark the function as variadic, allowing it to receive a variable number of parameters:

```
Join(sep, params*) {
 for index,param in params
 str .= param . sep
 return SubStr(str, 1, -StrLen(sep))
}
MsgBox Join("`n", "one", "two", "three")
```

When a variadic function is called, surplus parameters can be accessed via an object which is stored in the function's final parameter. The first surplus parameter is at `params[1]`, the second at `params[2]` and so on. As it is a standard `Object`, `params.Length()` can be used to determine the number of parameters.

Notes:

- The "variadic" parameter can only appear at the end of the formal parameter list.
- [RegEx callouts](#) cannot be variadic; the "variadic" parameter is tolerated but left blank.
- [Callbacks](#) pass surplus parameters [by address](#) rather than via an array.

## Variadic Function Calls

While variadic functions can *accept* a variable number of parameters, an array of parameters can be passed to *any* function by applying the same syntax to a function-call:

```
substrings := ["one", "two", "three"]
MsgBox % Join("`n", substrings*)
```

Notes:

- Numbering of parameters within the source array begins at 1.
- Optional parameters may be entirely omitted from the array.
- The array of parameters may contain named items when calling a user-defined function; in any other case, named items are not supported.
- The target function may also be variadic, in which case named items are copied even if they have no corresponding formal parameter.
- This syntax can also be used when calling methods or setting or retrieving properties of objects; for example, `Object.Property[Params*]`.

Known limitations:

- Only the right-most parameter can be expanded this way. For example, `Func(x, y*)` is supported but `Func(x*, y)` is not.
- There must not be any non-whitespace characters between the asterisk (\*) and the symbol which ends the parameter list.

# Local and Global Variables

## Local Variables

Local variables are specific to a single function and are visible only inside that function. Consequently, a local variable may have the same name as a global variable and both will have separate contents. Separate functions may also safely use the same variable names.

All local variables which are not `static` are automatically freed (made empty) when the function returns.

Variables access or created inside a function are local by default, with the following exceptions:

- Super-global variables, including classes.
- Built-in variables such as `Clipboard`, `ErrorLevel`, and `A_TimeIdle`.

The default may also be overridden as shown below.

## Global variables

To refer to an existing global variable inside a function (or create a new one), declare the variable as `global` prior to using it. For example:

```
LogToFile(TextToLog)
{
 global LogFileName ; This global variable
 was previously given a value somewhere outside
```

**this function.**

```
FileAppend TextToLog "`n", LogFileName
}
```

**Assume-global mode:** If a function needs to access or create a large number of global variables, it can be defined to assume that all its variables are global (except its parameters) by making its first line the word "global". For example:

```
SetDefaults()
{
 global
 MyGlobal := 33 ; Assigns 33 to a global
 variable, first creating the variable if
 necessary.
 local x, y:=0, z ; Local variables must be
 declared in this mode, otherwise they would be
 assumed global.
}
```

This assume-global mode can also be used by a function to create a global pseudo-array, such as a loop that assigns values to `Array%A_Index%`.

**Super-global variables:** If a global declaration appears outside of any function, it takes effect for all functions by default. This avoids the need to redeclare the variable in each function. However, if a function parameter or local variable with the same name is declared, it takes precedence over the global variable.

Variables created by the `class` keyword are also super-global.

## Static variables

Static variables are always implicitly local, but differ from locals because their values are remembered between calls. For example:

```
LogToFile(TextToLog)
{
 static LoggedLines := 0
 LoggedLines += 1 ; Maintain a tally
 locally (its value is remembered between
 calls).
 global LogFileName
 FileAppend LoggedLines ": " TextToLog "`n",
 LogFileName
}
```

A static variable may be initialized on the same line as its declaration by following it with `:=` followed by any `expression`. For example: `static X:=0, Y:="fox"`. Each static variable is initialized only once. Static variables are initialized in the order that they appear in the script file, but before the script's auto-execute section.

**Assume-static mode:** A function may be defined to assume that all its variables are static (except its parameters) by making its first line the word "static". For example:

```
GetFromStaticArray(WhichItemNumber)
{
 static
 static FirstCallToUs := true ; A static
 declaration's initializer still runs only once
 (upon startup).
 if FirstCallToUs ; Create a static array
```

during the first call, but not on subsequent calls.

```
{
 FirstCallToUs := false
 Loop 10
 StaticArray%A_Index% := "Value #" .
A_Index
 }
 return StaticArray%WhichItemNumber%
}
```

In assume-static mode, any variable that should not be static must be declared as local or global.

## More about locals and globals

Multiple variables may be declared on the same line by separating them with commas as in these examples:

```
global LogFileName, MaxRetries := 5
static TotalAttempts := 0, PrevResult
```

A local or global variable may be initialized on the same line as its declaration by following it with `:=` followed by any [expression](#). Unlike [static initializers](#), the initializers of locals and globals execute every time the function is called, but only if/when the flow of control actually reaches them. In other words, a line like `local x := 0` has the same effect as writing two separate lines: `local x` followed by `x := 0`.

Because the words *local*, *global*, and *static* are processed immediately when the

script launches, a variable cannot be conditionally declared by means of an **IF statement**. In other words, a declaration inside an IF's or ELSE's **block** takes effect unconditionally for all lines between the declaration and the function's closing brace. Also note that it is not currently possible to declare a dynamic variable such as `global Array%i%`.

## Dynamically Calling a Function

A function (even a [built-in function](#)) may be called dynamically via percent signs. For example, `%Var%(x, "fox")` would call the function whose name is contained in *Var*. Similarly, `Func%A_Index%()` would call `Func1()` or `Func2()`, etc., depending on the current value of `A_Index`.

*Var* in `%Var%()` can contain a function name or a [function object](#). If the function does not exist, the [default base object's](#) `Call` method is called instead (and as usual, if this method is undefined, the `__Call` meta-function is invoked).

If the function cannot be called due to one of the reasons below, an [exception](#) is thrown:

- Calling a nonexistent function, which can be avoided by using `If IsFunc(VarContainingFuncName)`. Except for [built-in functions](#), the called function's [definition](#) must exist explicitly in the script by means such as `#include` or a non-dynamic call to a [library function](#).
- Passing too few parameters, which can be avoided by checking `IsFunc()`'s return value (which is the number of mandatory parameters plus one). Note: Passing too many parameters is tolerated; each extra parameter is fully evaluated (including any calls to functions) and then discarded.

Finally, a dynamic call to a function is slightly slower than a normal call because normal calls are resolved (looked up) before the script begins running.

## Short-circuit Boolean Evaluation

When *AND*, *OR*, and the [ternary operator](#) are used within an [expression](#), they short-circuit to enhance performance (regardless of whether any function calls are present). Short-circuiting operates by refusing to evaluate parts of an expression that cannot possibly affect its final result. To illustrate the concept, consider this example:

```
if (ColorName <> "" AND not
 FindColor(ColorName))
 MsgBox %ColorName% could not be found.
```

In the example above, the `FindColor()` function never gets called if the *ColorName* variable is empty. This is because the left side of the *AND* would be *false*, and thus its right side would be incapable of making the final outcome *true*.

Because of this behavior, it's important to realize that any side-effects produced by a function (such as altering a global variable's contents) might never occur if that function is called on the right side of an *AND* or *OR*.

It should also be noted that short-circuit evaluation cascades into nested *ANDs* and *ORs*. For example, in the following expression, only the leftmost comparison occurs whenever *ColorName* is blank. This is because the left side would then be enough to determine the final answer with certainty:

```
if (ColorName = "" OR FindColor(ColorName,
```

```
Region1) OR FindColor(ColorName, Region2)
break ; Nothing to search for, or a match
was found.
```

As shown by the examples above, any expensive (time-consuming) functions should generally be called on the right side of an *AND* or *OR* to enhance performance. This technique can also be used to prevent a function from being called when one of its parameters would be passed a value it considers inappropriate, such as an empty string.

The [ternary conditional operator](#) (`?:`) also short-circuits by not evaluating the losing branch.

## Using Subroutines Within a Function

Although a function cannot contain [definitions](#) of other functions, it can contain subroutines. As with other subroutines, use [Gosub](#) to launch them and [Return](#) to return (in which case the [Return](#) would belong to the [Gosub](#) and not the function).

If a function uses [Gosub](#) to jump to a public subroutine (one that lies outside of the function's braces), all variables outside are global and the function's own [local variables](#) are not accessible until the subroutine returns. However, `A_ThisFunc` will still contain the name of the function.

Although [Goto](#) cannot be used to jump from inside a function to outside, it is possible for a function to [Gosub](#) an external/public subroutine and then do a [Goto](#) from there.

Although the use of [Goto](#) is generally discouraged, it can be used inside a function to jump to another position within the same function. This can help simplify complex functions that have many points of return, all of which need to do some clean-up prior to returning.

A function may contain externally-called subroutines such as [timers](#), and [menu items](#). This is generally done to encapsulate them in a separate file for use with `#Include`, which prevents them from interfering with the script's [auto-execute section](#). However, the following limitations apply:

- Such subroutines should use only [static](#) and [global](#) variables (not [locals](#)) if

their function is ever called normally. This is because a subroutine [thread](#) that interrupts a function-call thread (or vice versa) would be able to change the values of local variables seen by the interrupted thread. Furthermore, any time a function returns to its caller, all of its local variables are made blank to free their memory.

- When a function is entered by a subroutine [thread](#), any references to [dynamic variables](#) made by that thread are treated as [globals](#) (including commands that create arrays). Similarly, [local labels](#) cannot be referenced dynamically.

## Return, Exit, and General Remarks

If the flow of execution within a function reaches the function's closing brace prior to encountering a `Return`, the function ends and returns a blank value (empty string) to its caller. A blank value is also returned whenever the function explicitly omits `Return`'s parameter.

When a function uses the `Exit` command to terminate the `current thread`, its caller does not receive a return value at all. For example, the statement `Var := Add(2, 3)` would leave `Var` unchanged if `Add()` exits. The same thing happens if a function causes a runtime error such as `running` a nonexistent file (when `UseErrorLevel` is not in effect).

A function may alter the value of `ErrorLevel` for the purpose of returning an extra value that is easy to remember.

To call a function with one or more blank values (empty strings), use an empty pair of quotes as in this example: `FindColor(ColorName, "")`.

Since calling a function does not start a new `thread`, any changes made by a function to settings such as `SendMode` and `SetTitleMatchMode` will go into effect for its caller too.

The caller of a function may pass a nonexistent variable or `array` element to it, which is useful when the function expects the corresponding parameter to be `ByRef`. For example, calling `GetNextLine(BlankArray%i%)` would create the variable `BlankArray%i%` automatically as a `local` or `global`

(depending on whether the caller is inside a function and whether it has the `assume-global mode` in effect).

When used inside a function, `ListVars` displays a function's `local variables` along with their contents. This can help debug a script.

## Style and Naming Conventions

You might find that complex functions are more readable and maintainable if their special variables are given a distinct prefix. For example, naming each parameter in a function's parameter list with a leading "p" or "p\_" makes their special nature easy to discern at a glance, especially when a function has several dozen [local variables](#) competing for your attention. Similarly, the prefix "r" or "r\_" could be used for [ByRef](#) parameters, and "s" or "s\_" could be used for [static variables](#).

The [One True Brace \(OTB\)](#) style may optionally be used to define functions. For example:

```
Add(x, y) {
 return x + y
}
```

## Using #Include to Share Functions Among Multiple Scripts

The `#include` directive may be used (*even at the top of a script*) to load functions from an external file.

Explanation: When the script's flow of execution encounters a function definition, it jumps over it (using an instantaneous method) and resumes execution at the line after its closing brace. Consequently, execution can never fall into a function from above, nor does the presence of one or more functions at the very top of a script affect the `auto-execute` section.

## Libraries of Functions: Standard Library and User Library

A script may call a function in an external file without having to use `#Include`. For this to work, a file of the same name as the function must exist in one of the following library directories:

```
%A_ScriptDir%\Lib\ ; Local library.
%A_MyDocuments%\AutoHotkey\Lib\ ; User
library.
path-to-the-currently-running-
AutoHotkey.exe\Lib\ ; Standard library.
```

For example, if a script calls a nonexistent function `MyFunc()`, the program searches for a file named "MyFunc.ahk" in the user library. If not found there, it searches for it in the standard library. If a match is still not found and the function's name contains an underscore (e.g. `MyPrefix_MyFunc`), the program searches both libraries for a file named `MyPrefix.ahk` and loads it if it exists. This allows `MyPrefix.ahk` to contain both the function `MyPrefix_MyFunc` and other related functions whose names start with `MyPrefix_`.

The local library is searched before the user library and standard library.

Only a direct function call such as `MyFunc()` can cause a library to be auto-included. If the function is only called dynamically or indirectly, such as by a timer or GUI event, the library must be explicitly included in the script. For

example: `#Include <MyFunc>`

Although a library file generally contains only a single function of the same name as its filename, it may also contain private functions and subroutines that are called only by it. However, such functions should have fairly distinct names because they will still be in the global namespace; that is, they will be callable from anywhere in the script.

If a library file uses `#Include`, the working directory for `#Include` is the library file's own directory. This can be used to create a redirect to a larger library file that contains that function and others related to it.

The [script compiler \(ahk2exe\)](#) also supports library functions. However, it requires that a copy of `AutoHotkey.exe` exist in the directory above the compiler directory (which is normally the case). If `AutoHotkey.exe` is absent, the compiler still works but library functions are not automatically included.

Functions included from a library perform just as well as other functions because they are pre-loaded before the script begins executing.

## Built-in Functions

Any optional parameters at the end of a built-in function's parameter list may be completely omitted. For example, `WinExist("Untitled - Notepad")` is valid because its other three parameters would be considered blank.

A built-in function is overridden if the script defines its own function of the same name. For example, a script could have its own custom `WinExist()` function that is called instead of the standard one. However, the script would then have no way to call the original function.

External functions that reside in DLL files may be called with `DllCall()`.

To get more details about a particular built-in function below, simply click on its name.

## Macro Functions

A function definition can be prefixed with 'macro ' (followed by space, not tab!) to create a macro instead of function.

Macro will use caller's scope for all variables except for parameters, function parameters will be resolved to local variables.

Dynamic variable references will also use caller's scope.

Global and static variables are not supported in macros.

Deref function example:

```
fun()
fun(){
 MyVar:="AutoHotkey_H"
 MsgBox deref("Calling deref ``% from
%A_ThisFunc% in %MyVar%")
}
macro deref(__deref__, __out__:= "", __set__:= ""){
 Loop Parse, __deref__, "%"
 If __set__ && !__set__ := false
 __out__ .= %A_LoopField%
 else if
SubStr(A_LoopField, -1) = "`"
 __out__ .= SubStr(A_LoopField, 1, -1) "%"
 else
 __out__ .= A_LoopField, __set__ := true
 return __out__
}
```

## Alternative deref function example:

```
fun()
fun(){
 MyVar:="AutoHotkey_H"
 MsgBox
 deref(["A_ThisFunc", "MyVar"], "Calling deref ``%
 from `a in `a", "`a")
}
macro
deref(__vars__, __text__, __delimiter__, __out__:=
""){
 Loop Parse, __text__, __delimiter__
 if A_LoopField
__delimiter__:=__vars__[A_Index], __out__.+=A_Loo
pField %__delimiter__%
 return __out__
}
```

## Frequently-used Functions

| Function    | Description                                                              |
|-------------|--------------------------------------------------------------------------|
| DirExist    | Checks for the existence of a folder and returns its attributes.         |
| FileExist   | Checks for the existence of a file or folder and returns its attributes. |
| GetKeyState | Returns true (1) if the specified key is down and false (0) if it is up. |
| InStr       | Searches for a given occurrence of a string, from the left or the right. |
|             | Determines whether a string contains a pattern (regular                  |

|              |                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------|
| RegExMatch   | expression).                                                                                                |
| RegExReplace | Replaces occurrences of a pattern (regular expression) inside a string.                                     |
| StrLen       | Retrieves the count of how many characters are in a string.                                                 |
| StrReplace   | Replaces occurrences of the specified substring with a new string.                                          |
| StrSplit     | Separates a string into an array of substrings using the specified delimiters.                              |
| SubStr       | Retrieves one or more characters from the specified position in a string.                                   |
| WinActive    | Checks if the specified window exists and is currently active (foremost), and returns its Unique ID (HWND). |
| WinExist     | Checks if a matching window exists and returns the Unique ID (HWND) of the first matching window.           |

## Miscellaneous Functions

| Function         | Description                                                                                                            |
|------------------|------------------------------------------------------------------------------------------------------------------------|
| Chr              | Returns the string (usually a single character) corresponding to the character code indicated by the specified number. |
| DllCall          | Calls a function inside a DLL, such as a standard Windows API function.                                                |
| FileOpen         | Provides object-oriented file I/O.                                                                                     |
| Func             | Retrieves a reference to the specified function.                                                                       |
| GetKeyName/VK/SC | Retrieves the name/text, virtual key code or scan code of a key.                                                       |
| IsByRef          | Returns a non-zero number if a ByRef parameter of a                                                                    |

|                  |                                                                                                    |
|------------------|----------------------------------------------------------------------------------------------------|
|                  | function was supplied with the specified variable.                                                 |
| IsFunc           | Returns a non-zero number if the specified function exists in the script.                          |
| IsLabel          | Returns a non-zero number if the specified label exists in the script.                             |
| IsObject         | Returns a non-zero number if the specified value is an object.                                     |
| ListView         | Functions to add, insert, modify or delete ListView rows/columns, or to get data from them.        |
| NumGet           | Returns the binary number stored at the specified address+offset.                                  |
| NumPut           | Stores a number in binary format at the specified address+offset.                                  |
| OnMessage        | Monitors a message/event.                                                                          |
| Ord              | Returns the ordinal value (numeric character code) of the first character in the specified string. |
| StrGet           | Copies a string from a memory address, optionally converting it between code pages.                |
| StrPut           | Copies a string to a memory address, optionally converting it between code pages.                  |
| RegisterCallback | Creates a machine-code address that when called, redirects the call to a function in the script.   |
| TreeView         | Functions to add, modify or delete TreeView items, or to get data from them.                       |
| Trim             | Trims characters from the beginning and/or end of a string.                                        |
| VarSetCapacity   | Enlarges a variable's holding capacity or frees its memory.                                        |

| Function              | Description                                                                                 |
|-----------------------|---------------------------------------------------------------------------------------------|
| Abs                   | Returns the absolute value of <i>Number</i> .                                               |
| Ceil                  | Returns <i>Number</i> rounded up to the nearest integer (without any .00 suffix).           |
| Exp                   | Returns <i>e</i> (which is approximately 2.71828182845905) raised to the <i>N</i> th power. |
| Floor                 | Returns <i>Number</i> rounded down to the nearest integer (without any .00 suffix).         |
| Log                   | Returns the logarithm (base 10) of <i>Number</i> .                                          |
| Ln                    | Returns the natural logarithm (base <i>e</i> ) of <i>Number</i> .                           |
| Mod                   | Returns the remainder when <i>Dividend</i> is divided by <i>Divisor</i> .                   |
| Round                 | Returns <i>Number</i> rounded to <i>N</i> decimal places.                                   |
| Sqrt                  | Returns the square root of <i>Number</i> .                                                  |
| Sin / Cos /<br>Tan    | Returns the trigonometric sine/cosine/tangent of <i>Number</i> .                            |
| ASin / ACos /<br>ATan | Returns the arcsine/arccosine/arctangent in radians.                                        |

# Debugger Clients

Additional debugging features are supported via [DBGp](#), a common debugger protocol for languages and debugger UI communication. See [Interactive Debugging](#) for more details. Some UIs or "clients" known to be compatible with AutoHotkey are listed on this page.

## SciTE4AutoHotkey

SciTE4AutoHotkey is a free, SciTE-based AutoHotkey script editor. In addition to DBGp support, it provides syntax highlighting, calltips/parameter info and auto-complete for AutoHotkey, and other useful editing features and scripting tools.

Debugging features include:

- Breakpoints.
- Run, Step Over/Into/Out.
- View the call stack.
- List name and contents of variables in local/global scope.
- Hover over variable to show contents.
- Inspect or edit variable contents.
- View structure of objects.

<http://fincs.abk4.net/scite4ahk/>

## XDebugClient

XDebugClient is a simple open-source front-end DBGp client based on the **.NET Framework 2.0**. XDebugClient was originally designed for PHP with Xdebug, but a custom build compatible with AutoHotkey is available below.

### Changes:

- Allow the debugger engine to report a language other than "php".
- Added AutoHotkey syntax highlighting.
- Automatically listen for a connection from the debugger engine, rather than waiting for the user to click *Start Listening*.
- Truncate property values at the first null-character, since AutoHotkey currently returns the entire variable contents and XDebugClient has no suitable interface for displaying binary content.

**Download:** [Binary](#); [Source Code](#) (also see [SharpDevelop](#), [Dockpanel Suite](#) and [Advanced Treeview](#).)

### Usage:

- Launch XDebugClient.
- Launch AutoHotkey /Debug. XDebugClient should automatically open the script file.
- Click the left margin to set at least one breakpoint.
- Choose Run from the Debug menu, or press F5.
- When execution hits a breakpoint, use the Debug menu or shortcut keys to

step through or resume the script.

## Features:

- Syntax highlighted, read-only view of the source code.
- Breakpoints.
- Run, Step Over/Into/Out.
- View the call stack.
- Inspect variables - select a variable name, right-click, Inspect.

## Issues:

- The user interface does not respond to user input while the script is running.
- No mechanisms are provided to list variables or set their values.

## Notepad++ DBGp Plugin

A DBGp client is available as a plugin for [Notepad++](#). It is designed for PHP, but also works well with [AutoHotkey](#).

**Download:** See [Directory for Notepad++ plugins](#).

### Usage:

- Launch Notepad++.
- Configure the DBGp plugin via *Plugins, DBGp, Config...*

**Note:** File Mapping must be configured. Most users will not be debugging remotely, and therefore may simply put a checkmark next to *Bypass all mapping (local windows setup)*.

- Show the debugger pane via the toolbar or **Plugins, DBGp, Debugger**.
- Open the script file to be debugged.
- Set at least one breakpoint.
- Launch AutoHotkey /Debug.
- Use the debugger toolbar or shortcut keys to control the debugger.

### Features:

- Syntax highlighting, if configured by the user.
- Breakpoints.
- Run, Step Over/Into/Out, Run to cursor, Stop.
- View local/global variables.

- Watch user-specified variables.
- View the call stack.
- Hover over a variable to view its contents.
- User-configurable shortcut keys - Settings, Shortcut Mapper..., Plugin commands.

## Issues:

- Hovering over a single-letter variable name does not work - for instance, hovering over "a" will attempt to retrieve " a" or "a ".
- Hovering over text will attempt to retrieve a variable even if the text contains invalid characters.
- Notepad++ becomes unstable if `property_get` fails, which is particularly problematic in light of the above. As a workaround, AutoHotkey sends an empty property instead of an error code when a non-existent or invalid variable is requested.

## Script-based Clients

A script-based DBGp library and example clients are available from GitHub.

- `dbg_console.ahk`: Simple command-line client.
- `dbg_test.ahk`: Demonstrates asynchronous debugging.
- `dbg_listvars.ahk`: Example client which just lists the variables of all running scripts.

GitHub: [Lexikos / dbg](#)

## Command-line Client

A command-line client is available from [xdebug.org](http://xdebug.org), however this is not suitable for most users as it requires a decent understanding of DBGp (the protocol).

## Others

A number of other DBGp clients are available, but have not been tested with AutoHotkey. For a list, see [Xdebug: Documentation](#).

# Objects

An *object* in AutoHotkey is an abstract datatype which provides three basic functions:

- GET a value.
- SET a value.
- CALL a method (that is, a function which does something with the target object).

An object *reference* is a pointer or "handle" to a particular object. Like strings and numbers, object references can be stored in variables, passed to or returned from functions and stored in objects. After copying a reference from one variable to another as in `x := y`, both variables refer to the same object.

**IsObject** can be used to determine if a value is an object:

```
Result := IsObject(expression)
```

Types of objects include:

- **Object** - scriptable associative array.
- **File** - provides an interface for file input/output.
- **Function objects** - **Func**, **BoundFunc** or **user-defined**.
- **ComObject** - wraps an IDispatch interface (a COM or "Automation" object).
- **DynaCall** - wraps an exported function in a callable object.

- `Struct` - creates a structure object.

## Table of Contents

- Basic Usage - Simple Arrays, Associative Arrays, Objects, Freeing Objects, Remarks
- Extended Usage - Function References, Arrays of Arrays, Arrays of Functions
- Custom Objects - Prototypes, Classes, Construction and Destruction, Meta-Functions
- Default Base Object - Automatic Var Init, Pseudo-Properties, Debugging
- Implementation - Reference-Counting, Pointers to Objects

## Basic Usage

### Simple Arrays

Create an array:

```
Array := [Item1, Item2, ..., ItemN]
Array := Array(Item1, Item2, ..., ItemN)
```

Retrieve an item:

```
Value := Array[Index]
```

Assign an item:

```
Array[Index] := Value
```

Insert one or more items at a given index:

```
Array.InsertAt(Index, Value, Value2, ...)
```

Append one or more items:

```
Array.Push(Value, Value2, ...)
```

Remove an item:

```
RemovedValue := Array.RemoveAt(Index)
```

Remove the last item:

```
RemovedValue := Array.Pop()
```

If the array is not empty, `MinIndex` and `MaxIndex/Length` return the lowest and highest index currently in use in the array. Since the lowest index is nearly always 1, `MaxIndex` usually returns the number of items. However, if there are no integer keys, `MaxIndex` returns an empty string whereas `Length` returns 0. Looping through an array's contents can be done either by index or with a `For`-loop. For example:

```
array := ["one", "two", "three"]

; Iterate from 1 to the end of the array:
Loop array.Length()
 MsgBox array[A_Index]

; Enumerate the array's contents:
For index, value in array
 MsgBox "Item " index " is '" value "'"
```

## Associative Arrays

An associative array is an object which contains a collection of unique keys and a collection of values, where each key is associated with one value. Keys can be strings, integers or objects, while values can be of any type. An associative array

can be created as follows:

```
Array := {KeyA: ValueA, KeyB: ValueB, ...,
KeyZ: ValueZ}
Array := Object("KeyA", ValueA, "KeyB", ValueB,
..., "KeyZ", ValueZ)
```

Using the `{key:value}` notation, quote marks are optional for keys which consist only of word characters. Any expression can be used as a key, but to use a variable as a key, it must be enclosed in parentheses. For example,

`{(KeyVar): Value}` and `{GetKey(): Value}` are both valid.

Retrieve an item:

```
Value := Array[Key]
```

Assign an item:

```
Array[Key] := Value
```

Remove an item:

```
RemovedValue := Array.Delete(Key)
```

Enumerating items:

```
array := {ten: 10, twenty: 20, thirty: 30}
For key, value in array
 MsgBox key ' = ' value
```

---

Associative arrays can be sparsely populated - that is, `{1:"a",1000:"b"}` contains only two key-value pairs, not 1000.

In AutoHotkey v1.x, simple arrays and associative arrays are the same thing. However, treating `arr` as a simple linear array helps to keep its role clear, and improves the chance of your script working with a future version of AutoHotkey, which might differentiate between simple arrays and associative arrays.

## Objects

For all types of objects, the notation `Object.LiteralKey` can be used to access a property, array element or method, where *LiteralKey* is an identifier or integer and *Object* is any expression. Identifiers are unquoted strings which may consist of alphanumeric characters, underscore and non-ASCII characters. For example, `match.Pos` is equivalent to `match["Pos"]` while `arr.1` is equivalent to `arr[1]`. There must be no space after the dot.

### Examples:

Retrieve a property:

```
Value := Object.Property
```

Set a property:

```
Object.Property := Value
```

Call a method:

```
ReturnValue := Object.Method(Parameters)
```

Call a method with a computed method name:

```
ReturnValue := Object[MethodName](Parameters)
```

Some properties of COM objects and user-defined objects can accept parameters:

```
Value := Object.Property[Parameters]
Object.Property[Parameters] := Value
```

**Related:** [Object](#), [File Object](#), [Func Object](#), [COM object](#)

### Known limitation:

- Currently `x.y[z]()` is treated as `x["y", z]()`, which is not supported. As a workaround, `(x.y)[z]()` evaluates `x.y` first, then uses the result as the target of the method call. Note that `x.y[z].Call()` does not have this limitation since it is evaluated the same as `(x.y[z]).Call()`.

## Freeing Objects

Scripts do not free objects explicitly. When the last reference to an object is released, the object is freed automatically. A reference stored in a variable is

released automatically when that variable is assigned some other value. For example:

```
obj := {} ; Creates an object.
obj := "" ; Releases the last reference, and
therefore frees the object.
```

Similarly, a reference stored in a field of another object is released when that field is assigned some other value or removed from the object. This also applies to arrays, which are actually objects.

```
arr := [{}] ; Creates an array containing an
object.
arr[1] := {} ; Creates a second object,
implicitly freeing the first object.
arr.RemoveAt(1) ; Removes and frees the second
object.
```

Because all references to an object must be released before the object can be freed, objects containing circular references aren't freed automatically. For instance, if `x.child` refers to `y` and `y.parent` refers to `x`, clearing `x` and `y` is not sufficient since the parent object still contains a reference to the child and vice versa. To resolve this situation, remove the circular reference.

```
x := {}, y := {} ; Create two
objects.
x.child := y, y.parent := x ; Create a
circular reference.

y.parent := "" ; The circular
```

reference must be removed before the objects can be freed.

```
x := "", y := "" ; Without the
above line, this would not free the objects.
```

For more advanced usage and details, see [Reference Counting](#).

## Remarks

## Syntax

All types of objects support both array syntax (brackets) and object syntax (dots).

Additionally, object references can themselves be used in expressions:

- When an object reference is compared with some other value using one of `=`, `==`, `!=`, `<>`, they are considered equal only if both values are references to the same object.
- Objects are always considered *true* when a boolean value is required, such as in `if obj`, `!obj` or `obj ? x : y`.
- An object's address can be retrieved using the `&` address-of operator. This uniquely identifies the object from the point of its creation to the moment its last reference is [released](#).

If an object is used in any context where an object is not expected, it is treated as an empty string. For example, `MsgBox myObject` shows an empty MsgBox and `myObject + 1` yields an empty string. Do not rely on this behaviour as it may change.

When a method-call is followed immediately by an assignment operator, it is equivalent to setting a property with parameters. For example, the following are equivalent:

```
obj.item(x) := y
obj.item[x] := y
```

Compound assignments such as `x.y += 1` and `--arr[1]` are supported.

Parameters can be omitted when getting or setting properties. For example, `x[, 2]`. Scripts can utilize this by defining default values for parameters in [properties](#) and [meta-functions](#). The method name can also be completely omitted, as in `x[](a)`. Scripts can utilize this by defining a default value for the `__Call` [meta-function](#)'s first parameter, since it is not otherwise supplied with a value. If the property or method name is omitted when invoking a COM object, its "default member" is invoked.

## Keys

Objects created with `[]`, `{}` or the `new` operator allow the use of strings, integers and objects as keys, with the following caveats:

The key's value is preserved, but its *type identity* is not. That is, integers may be stored as strings or vice versa, so long as the value remains the same (including the formatting of numeric strings). Specifically:

- Integer keys are stored using the native signed integer type where possible. Integers which are less than -2147483648 or greater than 2147483647 are

stored as strings on AutoHotkey 32-bit but as integers on AutoHotkey 64-bit. (By contrast, 64-bit integers can be stored as values on either version.)

- If a string key can be converted to an integer (of the native signed integer type) and back to a string without loss of data, it is stored as an integer. In other words, `x[16]` and `x["16"]` are equivalent, but not `x["0016"]` or `x["0x10"]`. However, since numeric literals are converted to pure numbers at load-time, `x[0x10]` is equivalent to `x["16"]`.
- Floating-point numbers are not supported as keys - instead they are converted to strings. For consistency and clarity, scripts should avoid using floating-point literals as keys.

Some strings generally should not be used as keys:

- By default, the string key "base" is used to retrieve or set the object's [base object](#), so cannot be used for storing ordinary values with a normal assignment. However, if a value is stored by some other means (such as `ObjRawSet(Object, "base", "")` or `Object.SetCapacity("base", 0)`), the key "base" then acts like any other string.
- Although [built-in method](#) names such as "Length" can be used as keys, storing a value will prevent the corresponding method from being called (unless that value is a reference to the appropriate function, such as `ObjLength`).

## Extended Usage

### Function References

If the variable *func* contains a function name, the function can be called with the expression `%Func%()`. However, this requires the function name to be resolved each time, which is inefficient if the function is called more than once. To improve performance, the script can retrieve a reference to the function and store it for later use:

```
MyFuncRef := Func("MyFunc")
```

A function can be called by reference using the following syntax:

```
RetVal := %MyFuncRef% (Params)
RetVal := MyFuncRef.Call(Params)
```

For details about additional properties of function references, see [Func Object](#).

### Arrays of Arrays

AutoHotkey supports "multi-dimensional" arrays by transparently storing arrays inside other arrays. For example, a table could be represented as an array of rows, where each row is itself an array of columns. In that case, the content of column `y` of row `x` can be set using either of the methods below:

```
table[x][y] := content ; A
```

```
table[x, y] := content ; B
```

If `table[x]` does not exist, *A* and *B* differ in two ways:

- *A* fails whereas *B* automatically creates an object and stores it in `table[x]`.
- If `table`'s `base` defines `meta-functions`, they are invoked as follows:

```
table.base.__Get(table, x)[y] := content ;
A
table.base.__Set(table, x, y, content) ;
B
```

Consequently, *B* allows the object to define custom behaviour for the overall assignment.

Multi-dimensional assignments such as `table[a, b, c, d] := value` are handled as follows:

- If there is only one key remaining, perform the assignment and return. Otherwise:
- Search the object for the first key in the list.
- If a non-object is found, fail (throw an exception).
- If an object is not found, create one and store it.
- Recursively invoke the sub-object, passing the remaining keys and value - repeat from the top.

This behaviour only applies to script-created objects, not more specialized types of objects such as COM objects or COM arrays.

## Arrays of Functions

An array of functions is simply an array containing function names or references. For example:

```
array := [Func("FirstFunc"),
Func("SecondFunc")]

; Call each function, passing "foo" as a
parameter:
Loop 2
 array[A_Index].Call("foo")

; Call each function, implicitly passing the
array itself as a parameter:
Loop 2
 array[A_Index]()

FirstFunc(param) {
 MsgBox A_ThisFunc ": " (IsObject(param) ?
"object" : param)
}
SecondFunc(param) {
 MsgBox A_ThisFunc ": " (IsObject(param) ?
"object" : param)
}
```

## Custom Objects

Objects created by the script do not need to have any predefined structure. Instead, each object can inherit properties and methods from its `base` object (otherwise known as a "prototype" or "class"). Properties and methods can also be added to (or removed from) an object at any time, and those changes will affect any and all derived objects. For more complex or specialized situations, a base object can override the standard behaviour of any objects derived from it by defining *meta-functions*.

Base objects are just ordinary objects, and are typically created one of two ways:

```
class baseObject {
 static foo := "bar"
}
; OR
baseObject := {foo: "bar"}
```

To create an object derived from another object, scripts can assign to the `base` property or use the `new` keyword:

```
obj1 := Object(), obj1.base := baseObject
obj2 := {base: baseObject}
obj3 := new baseObject
MsgBox obj1.foo " " obj2.foo " " obj3.foo
```

It is possible to reassign an object's `base` at any time, effectively replacing all of the properties and methods that the object inherits.

## Prototypes

Prototype or **base** objects are constructed and manipulated the same as any other object. For example, an ordinary object with one property and one method might be constructed like this:

```
; Create an object.
thing := {}
; Store a value.
thing.foo := "bar"
; Create a method by storing a function
reference.
thing.test := Func("thing_test")
; Call the method.
thing.test()

thing_test(this) {
 MsgBox this.foo
}
```

When `thing.test()` is called, *thing* is automatically inserted at the beginning of the parameter list. By convention, the function is named by combining the "type" of object and the method name.

An object is a *prototype* or *base* if another object derives from it:

```
other := {}
other.base := thing
other.test()
```

In this case, *other* inherits *foo* and *test* from *thing*. This inheritance is dynamic,

so if `thing.foo` is modified, the change will be reflected by `other.foo`. If the script assigns to `other.foo`, the value is stored in *other* and any further changes to `thing.foo` will have no effect on `other.foo`. When `other.test()` is called, its *this* parameter contains a reference to *other* instead of *thing*.

## Classes

At its root, a "class" is a set or category of things having some property or attribute in common. Since a [base](#) or [prototype](#) object defines properties and behaviour for set of objects, it can also be called a *class* object. For convenience, base objects can be defined using the "class" keyword as shown below:

```
class ClassName extends BaseClassName
{
 InstanceVar := Expression
 static ClassVar := Expression

 class NestedClass
 {
 ...
 }

 Method()
 {
 ...
 }

 Property[] ; Brackets are optional
 {
 get {
 return ...
 }
 }
}
```

```
 }
 set {
 return ... := value
 }
}
}
```

When the script is loaded, this constructs an object and stores it in the [super-global](#) variable `ClassName`. If `extends BaseClassName` is present, `BaseClassName` must be the full name of another class, but the order that they are defined in does not matter. The full name of each class is stored in `object.___Class`.

Within this documentation, the word "class" on its own usually means a class object constructed with the `class` keyword.

Class definitions can contain variable declarations, method definitions and nested class definitions.

## Instance Variables

An *instance variable* is one that each instance of the class (that is, each object derived from the class) has its own copy of. They are declared like normal assignments, but the `this.` prefix is omitted (only directly within the class body):

```
InstanceVar := Expression
```

These declarations are evaluated each time a new instance of the class is created

with the `new` keyword. The method name `__Init` is reserved for this purpose, and should not be used by the script. The `__New()` method is called after all such declarations have been evaluated, including those defined in base classes.

*Expression* can access other instance variables and methods via `this`, but all other variable references are assumed to be global.

To access an instance variable (even within a method), always specify the target object; for example, `this.InstanceVar`.

Declarations like `x.y := z` are also supported, provided that `x` was previously declared in this class. For example, `x := {}, x.y := 42` declares `x` and also initializes `this.x.y`.

## Static/Class Variables

Static/class variables belong to the class itself, but can be inherited by derived objects (including sub-classes). They are declared like instance variables, but using the `static` keyword:

```
static ClassVar := Expression
```

Static declarations are evaluated only once, before the `auto-execute` section, in the order they appear in the script. Each declaration stores a value in the class object. Any variable references in *Expression* are assumed to be global.

To assign to a class variable, always specify the class object; for example, `ClassName ClassVar := Value`. If an object `x` is derived from `ClassName` and `x` itself does not contain the key "ClassVar", `x.ClassVar` may

also be used to dynamically retrieve the value of `ClassName.ClassVar`. However, `x.ClassVar := y` would store the value in `x`, not in `ClassName`.

Declarations like `static x.y := z` are also supported, provided that `x` was previously declared in this class. For example, `static x := {}, x.y := 42` declares `x` and also initializes `ClassName.x.y`.

## Nested Classes

Nested class definitions allow a class object to be stored inside another class object rather than a separate global variable. In the example above, `class NestedClass` constructs an object and stores it in `ClassName.NestedClass`. Sub-classes could inherit `NestedClass` or override it with their own nested class (in which case `new this.NestedClass` could be used to instantiate whichever class is appropriate).

```
class NestedClass
{
 ...
}
```

## Methods

Method definitions look identical to function definitions. Each method has a hidden parameter named `this`, which typically contains a reference to an object derived from the class. However, it could contain a reference to the class itself or a derived class, depending on how the method was called. Methods are

stored [by reference](#) in the class object.

```
Method()
{
 ...
}
```

Inside a method, the pseudo-keyword `base` can be used to access the super-class versions of methods or properties which are overridden in a derived class. For example, `base.Method()` in the class defined above would call the version of *Method* which is defined by *BaseClassName*. [Meta-functions](#) are not called; otherwise, `base.Method()` behaves like `BaseClassName.Method.Call(this)`. That is,

- `base.Method()` always invokes the base of the class where the current method was defined, even if `this` is derived from a *sub-class* of that class or some other class entirely.
- `base.Method()` implicitly passes `this` as the first (hidden) parameter.

`base` only has special meaning if followed by a dot `.` or brackets `[]`, so code like `obj := base, obj.Method()` will not work. Scripts can disable the special behaviour of *base* by assigning it a non-empty value; however, this is not recommended.

## Properties

Property definitions allow a method to be executed whenever the script gets or sets a specific key.

```
Property[]
{
 get {
 return ...
 }
 set {
 return ... := value
 }
}
```

*Property* is simply the name of the property, which will be used to invoke it. For example, `obj.Property` would call *get* while `obj.Property := value` would call *set*. Within *get* or *set*, `this` refers to the object being invoked. Within *set*, `value` contains the value being assigned.

Parameters can be passed by enclosing them in square brackets to the right of the property name, both when defining the property and when calling it. Aside from using square brackets, parameters of properties are defined the same way as parameters of methods - optional, `ByRef` and variadic parameters are supported.

The return value of *get* or *set* becomes the result of the sub-expression which invoked the property. For example, `val := obj.Property := 42` stores the return value of *set* in `val`.

Each class can define one or both halves of a property. If a class overrides a property, it can use `base.Property` to access the property defined by its base class. If *get* or *set* is not defined, it can be handled by a base class. If *set* is not defined and is not handled by a meta-function or base class, assigning a value stores it in the object, effectively disabling the property.

Internally, *get* and *set* are two separate methods, so cannot share variables (except by storing them in `this`).

**Meta-functions** provide a broader way of controlling access to properties and methods of an object, but are more complicated and error-prone.

## Construction and Destruction

Whenever a derived object is created with the `new` keyword, the `__New` method defined by its base object is called. This method can accept parameters, initialize the object and override the result of the `new` operator by returning a value. When an object is destroyed, `__Delete` is called. For example:

```
m1 := new GMem(0, 20)
m2 := {base: GMem}.__New(0, 30)

class GMem
{
 __New(aFlags, aSize)
 {
 this.ptr := DllCall("GlobalAlloc",
"uint", aFlags, "ptr", aSize, "ptr")
 if !this.ptr
 return ""
 MsgBox "New GMem of " aSize " bytes at
address " this.ptr "."
 return this ; This line can be omitted
when using the 'new' operator.
 }

 __Delete()
 {
```

```
 MsgBox "Delete GMem at address "
this.ptr "."
 DllCall("GlobalFree", "ptr", this.ptr)
 }
}
```

`__Delete` is not called for any object which has the key "`__Class`". [Class objects](#) have this key by default.

## Meta-Functions

### Method syntax:

```
class ClassName {
 __Get([Key, Key2, ...])
 __Set([Key, Key2, ...], Value)
 __Call(Name [, Params...])
}
```

### Function syntax:

```
MyGet(this [, Key, Key2, ...])
MySet(this [, Key, Key2, ...], Value)
MyCall(this, Name [, Params...])
```

```
ClassName := { __Get: Func("MyGet"), __Set:
Func("MySet"), __Call: Func("MyCall") }
```

Meta-functions define what happens when a key is requested but not found within the target object. For example, if `obj.key` has not been assigned a value, it invokes the `__Get` meta-function. Similarly, `obj.key := value`

invokes `__Set` and `obj.key()` invokes `__Call`. These meta-functions (or methods) would need to be defined in `obj.base`, `obj.base.base` or such.

When the script gets, sets or calls a key which does not exist within the target object, the base object is invoked as follows:

- If this base object defines the appropriate meta-function, call it. If the meta-function explicitly `returns`, use the return value as the result of the operation (whatever caused the meta-function to be called) and return control to the script. Otherwise, continue as described below.

*Set*: If the meta-function handled an assignment, it should return the value which was assigned. This allows assignments to be chained, as in `a.x := b.y := z`. The return value may differ from the original value of `z` (for instance, if restrictions are imposed on which values can be assigned).

- Search for a matching key in the base object's own fields.
- If a key corresponding to a property is found and it implements *get* or *set* (as appropriate), invoke the property and return. If this is a method call, invoke *get*.
- If no key was found, recursively invoke this base object's own base (apply each of these steps to it, starting at the top of this list). If we're not finished yet, search this base object for a matching key again in case one was added by a meta-function.

Due to backward-compatibility, this step is performed for *set* operations even if a key was found (unless it defines a property which implements *set*).

- If multiple parameters were given for *get* or *set* and a key was found, check its value. If that value is an object, handle the remaining parameters by invoking it, and do nothing further.
- If a key was found, stop searching and do the following:
  - Get*: Return the value.
  - Set*: Carry out the default behaviour described below.
  - Call*: Attempt to call the value, passing the target object as the first parameter (`this`). The value should be a function name or a [function object](#).

If a meta-function stores a matching key in the object but does not `return`, the behaviour is the same as if the key initially existed in the object. For an example using `__Set`, see [Sub-classing Arrays of Arrays](#).

If the operation still hasn't been handled, check if this is a built-in method or property:

- *Get*: If the key is "base", return the object's base.
- *Set*: If the key is "base", set the object's base (or remove it if the value isn't an object).
- *Call*: Call a [built-in method](#) if applicable.

If the operation still hasn't been handled,

- *Get* and *Call*: Return an empty string.
- *Set*: If only one key parameter was given, store the key and value in the target object and return the assigned value. If multiple parameters were

given, create a new object and store it using the first parameter as a key, then handle the remaining parameters by invoking the new object. (See [Arrays of Arrays.](#))

### Known limitation:

- Using `return` without a value is equivalent to `return ""`. This may be changed in a future version so that `return` can be used to "escape" from a meta-function without overriding the default behaviour.

### Dynamic Properties

[Property syntax](#) can be used to define properties which compute a value each time they are evaluated, but each property must be known in advance and defined individually in the script. By contrast, `__Get` and `__Set` can be used to implement properties which aren't known by the script.

For example, a "proxy" object could be created which sends requests for properties over the network (or through some other channel). A remote server would send back a response containing the value of the property, and the proxy would return the value to its caller. Even if the name of each property was known in advance, it would not be logical to define each property individually in the proxy class since every property does the same thing (send a network request). Meta-functions receive the property name as a parameter, so are a good solution for this problem.

Another use of `__Get` and `__Set` is to implement a set of related properties which share code. In the example below they are used to implement a "Color" object

with R, G, B and RGB properties, where only the RGB value is actually stored:

```
red := new Color(0xff0000), red.R -= 5
cyan := new Color(0), cyan.G := 255, cyan.B :=
255

MsgBox "red: " red.R ", " red.G ", " red.B " = "
red.RGB
MsgBox "cyan: " cyan.R ", " cyan.G ", " cyan.B "
= " cyan.RGB

class Color
{
 __New(aRGB)
 {
 this.RGB := aRGB
 }

 static Shift := {R:16, G:8, B:0}

 __Get(aName)
 {
 ; NOTE: Using this.Shift here would
 cause an infinite loop!
 shift := Color.Shift[aName] ; Get the
 number of bits to shift.
 if (shift != "") ; Is it a known
 property?
 return (this.RGB >> shift) & 0xff
 ; NOTE: Using 'return' here would break
 this.RGB.
 }

 __Set(aName, aValue)
 {
 if ((shift := Color.Shift[aName]) !=
```

```

 """)
 {
 aValue &= 255 ; Truncate it to the
proper range.
 ; Calculate and store the new RGB
value.
 this.RGB := (aValue << shift) |
(this.RGB & ~(0xff << shift))

 ; 'Return' must be used to indicate
a new key-value pair should not be created.
 ; This also defines what will be
stored in the 'x' in 'x := clr[name] := val':
 return aValue
 }
 ; NOTE: Using 'return' here would break
this._RGB and this.RGB.
}

; Meta-functions can be mixed with
properties:
RGB {
 get {
 ; Return it in hex format:
 return format("0x{:06x}",
this._RGB)
 }
 set {
 return this._RGB := value
 }
}
}

```

However, in this case [Property syntax](#) could have been used instead, where code is shared by simply having each property call a central method. It is better to

avoid using meta-functions where possible due to the high risk of misuse (see the notes in red above).

## Objects as Functions

For an outline of how to create objects which can act as functions, see [Function Objects](#).

A function object can also act as a meta-function, such as to define dynamic properties similar to those in the previous section. Although it is recommended to use [property syntax](#) instead, the example below shows the potential of meta-functions for implementing new concepts or behaviour, or changing the structure of the script.

```
blue := new Color(0x0000ff)
MsgBox blue.R ", " blue.G ", " blue.B

class Properties
{
 Call(aTarget, aName, aParams*)
 {
 ; If this Properties object contains a
 definition for this half-property, call it.
 if ObjHasKey(this, aName)
 return this[aName].Call(aTarget,
aParams*)
 }
}

class Color
{
 __New(aRGB)
 {
```

```

 this.RGB := aRGB
 }

 class __Get extends Properties
 {
 R() {
 return (this.RGB >> 16) & 255
 }
 G() {
 return (this.RGB >> 8) & 255
 }
 B() {
 return this.RGB & 255
 }
 }
}

```

## Sub-classing Arrays of Arrays

When a [multi-parameter assignment](#) such as `table[x, y] := content` implicitly causes a new object to be created, the new object ordinarily has no base and therefore no custom methods or special behaviour. `__Set` may be used to initialize these objects, as demonstrated below.

```

x := {base: {addr: Func("x_Addr"), __Set:
Func("x_Setter")}}
; Assign value, implicitly calling x_Setter to
create sub-objects.
x[1,2,3] := "...".
; Retrieve value and call example method.

```

```

MsgBox x[1,2,3] "`n" x.addr() "`n" x[1].addr()
 "`n" x[1,2].addr()

x_Setter(x, p1, p2, p3) {
 x[p1] := new x.base
}

x_Addr(x) {
 return &x
}

```

Since `x_Setter` has four mandatory parameters, it will only be called when there are two or more key parameters. When the assignment above occurs, the following takes place:

- `x[1]` does not exist, so `x_Setter(x, 1, 2, 3)` is called (`"..."` is not passed as there are too few parameters).
  - `x[1]` is assigned a new object with the same base as `x`.
  - No value is returned – the assignment continues.
- `x[1][2]` does not exist, so `x_Setter(x[1], 2, 3, "...")` is called.
  - `x[1][2]` is assigned a new object with the same base as `x[1]`.
  - No value is returned – the assignment continues.
- `x[1][2][3]` does not exist, but since `x_Setter` requires four parameters and there are only three (`x[1][2], 3, "..."`), it is not called and the assignment completes as normal.

## Default Base Object

When a non-object value is used with object syntax, the *default base object* is invoked. This can be used for debugging or to globally define object-like behaviour for strings, numbers and/or variables. The default base may be accessed by using `.base` with any non-object value; for instance, `"".base`. Although the default base cannot be *set* as in `"".base := Object()`, the default base may itself have a base as in `"".base.base := Object()`.

## Automatic Var Init

When an empty variable is used as the target of a *set* operation, it is passed directly to the `__Set` meta-function, giving it opportunity to insert a new object into the variable. For brevity, this example does not support multiple parameters; it could, by using a [variadic function](#).

```
"".base.__Set :=
Func("Default_Set_AutomaticVarInit")

empty_var.foo := "bar"
MsgBox empty_var.foo

Default_Set_AutomaticVarInit(ByRef var, key,
value)
{
 if (var = "")
 var := Object(key, value)
}
```

## Pseudo-Properties

Object "syntax sugar" can be applied to strings and numbers.

```
"" .base.__Get :=
Func("Default_Get_PseudoProperty")
"" .base.is := Func("Default_is")

MsgBox A_AhkPath.length " == "
StrLen(A_AhkPath)
MsgBox A_AhkPath.length.is("int")

Default_Get_PseudoProperty(nonobj, key)
{
 if (key = "length")
 return StrLen(nonobj)
}

Default_is(nonobj, type)
{
 static alias := {int: "integer"}
 return nonobj is (alias[type] or type)
}
```

Note that built-in functions may also be used, but in this case the parentheses cannot be omitted:

```
"" .base.length := Func("StrLen")
MsgBox A_AhkPath.length() " == "
StrLen(A_AhkPath)
```

## Debugging

By default, invoking a non-object value causes an exception to be thrown. The following example changes the behaviour so that a warning is shown and the

script continues:

```
"".base.__Get := "".base.__Set :=
"".base.__Call := Func("Default__Warn")

empty_var.foo := "bar"
x := (1 + 1).is("integer")

Default__Warn(nonobj, p1="", p2="", p3="",
p4="")
{
 ListLines
 MsgBox "A non-object value was improperly
invoked.\n\nSpecifically: " nonobj
 return "" ; Override the default behaviour
by returning a value.
}
```

## Implementation

### Reference-Counting

AutoHotkey uses a basic reference-counting mechanism to automatically free the resources used by an object when it is no longer referenced by the script. Script authors should not invoke this mechanism explicitly, except when dealing directly with unmanaged [pointers to objects](#).

Temporary references returned by functions, methods or operators within an expression are released after evaluation of that expression has completed or been aborted. This allows temporary objects to be used for resource management. For example:

```
MsgBox DllCall("GlobalSize", "ptr", (new
GMem(0, 20)).ptr, "ptr") ; 20
```

To run code when the last reference to an object is being released, implement the [\\_\\_Delete](#) meta-function.

### Known Limitations:

- Circular references must be broken before an object can be freed. For details and an example, see [Freeing Objects](#).
- Although references in static and global variables are released automatically when the program exits, references in non-static local variables or on the expression evaluation stack are not. These references are

only released if the function or expression is allowed to complete normally.

Although memory used by the object is reclaimed by the operating system when the program exits, `__Delete` will not be called unless all references to the object are freed. This can be important if it frees other resources which are not automatically reclaimed by the operating system, such as temporary files.

## Pointers to Objects

In some rare cases it may be necessary to pass an object to external code via `DllCall` or store it in a binary data structure for later retrieval. An object's address can be retrieved via `address := &object`; however, this effectively makes two references to the object, but the program only knows about the one in *object*. If the last *known* reference to the object was released, the object would be deleted. Therefore, the script must inform the object that it has gained a reference. This can be done as follows:

```
ObjAddRef(address := &object)
```

The script must also inform the object when it is finished with that reference:

```
ObjRelease(address)
```

Generally each new copy of an object's address should be treated as another reference to the object, so the script should call `ObjAddRef` when it gains a copy and `ObjRelease` immediately before losing one. For example, whenever an address is copied via something like `x := address`, `ObjAddRef` should be

called. Similarly, when the script is finished with  $x$  (or is about to overwrite  $x$ 's value), it should call `ObjRelease`.

To convert an address to a proper reference, use `Object()`:

```
object := Object(address)
```

Note that the `Object()` function can be used even on objects which it did not create, such as [COM objects](#) and [File objects](#).

# Object

AutoHotkey's basic object datatype is an associative array with features which allow its behaviour to be [customized](#). By default, all objects created by `{}`, `[]`, `Object()` and `Array()` support the following methods and functions:

- [InsertAt / RemoveAt](#)
- [Push / Pop](#)
- [Delete](#)
- [MinIndex / MaxIndex / Length](#)
- [SetCapacity / GetCapacity](#)
- [GetAddress](#)
- [\\_NewEnum](#)
- [HasKey](#)
- [Clone](#)
- [ObjRawSet](#) (function)

Each method also has an equivalent function, which can be used to bypass any [custom behaviour](#) implemented by the object -- it is recommended that these functions only be used for that purpose. To call one, prefix the method name with "Obj" (but for `_NewEnum`, omit the underscore) and pass the target object as the first parameter. For example:

```
array := [1, 2, 3]
MsgBox % ObjLength(array) " = " array.Length()
```

# InsertAt

Inserts one or more values at a given position within a linear array.

```
Object.InsertAt(Pos, Value1 [, Value2, ... ValueN])
```

## Pos

The position to insert *Value1* at. Subsequent values are inserted at Pos+1, Pos+2, etc.

## Value1 ...

One or more values to insert. To insert an array of values, pass `theArray*` as the last parameter.

## Remarks

InsertAt is the counterpart of [RemoveAt](#).

As Objects are associative arrays, *Pos* is also the integer key which will be associated with *Value1*. Any items previously at or to the right of *Pos* are shifted to the right by the exact number of value parameters, even if some values are missing (i.e. the object is a sparse array). For example:

```
x := []
x.InsertAt(1, "A", "B") ; => ["A", "B"]
x.InsertAt(2, "C") ; => ["A", "C", "B"]

; Sparse/unassigned elements are preserved:
x := ["A", , "C"]
```

```
x.InsertAt(2, "B") ; => ["A", "B", ,
"C"]
```

```
x := ["C"]
x.InsertAt(1, , "B") ; => [, "B", "C"]
```

InsertAt should be used only when the object's integer keys represent positions in a linear array. If the object contains arbitrary integer keys such as IDs or handles, InsertAt is likely to cause unwanted side-effects. For example:

```
x := [], handleX := 0x4321, handleY := 0x1234
x.InsertAt(handleX, "A")
MsgBox % x[handleX] ; A - okay
x.InsertAt(handleY, "B")
MsgBox % x[handleX] ; Empty
MsgBox % x[handleX+1] ; This is the new
"position" of "B"
```

InsertAt does not affect string or object keys, so can be safely used with objects containing mixed key types.

# RemoveAt

Removes items from the given position in a linear array.

```
Object.RemoveAt(Pos [, Length])
```

## Pos

The position of the value or values to remove.

## Length

The length of the range of values to remove. Items from `Pos` to `Pos+Length-1` are removed. If omitted, one item is removed.

## Return Value

If *Length* is omitted, the value removed from *Pos* is returned (blank if none). Otherwise the return value is the number of removed items which had values, which can differ from *Length* in a sparse array, but is always between 0 and *Length* (inclusive).

## Remarks

RemoveAt is the counterpart of [InsertAt](#).

The remaining items to the right of *Pos* are shifted to the left by *Length* (or 1 if omitted), even if some items in the removed range did not have values. For example:

```
x := ["A", "B"]
MsgBox % x.RemoveAt(1) ; A
MsgBox % x[1] ; B
```

```
x := ["A", , "C"]
MsgBox % x.RemoveAt(1, 2) ; 1
MsgBox % x[1] ; C
```

RemoveAt should be used only when the object's integer keys represent positions in a linear array. If the object contains arbitrary integer keys such as IDs or handles, RemoveAt is likely to cause unwanted side-effects. For example:

```
x := {0x4321: "A", 0x1234: "B"}
MsgBox % x.RemoveAt(0x1234) ; B
MsgBox % x[0x4321] ; Empty
MsgBox % x[0x4321-1] ; A
```

RemoveAt does not affect string or object keys, so can be safely used with objects containing mixed key types.

# Push

Appends values to the end of an array.

```
Object.Push([Value, Value2, ..., ValueN])
```

## Value ...

One or more values to insert. To insert an array of values, pass `theArray*` as the last parameter.

## Return Value

The position of the last inserted value. Can be negative if the array only contained elements at negative indices.

## Remarks

The first value is inserted at position 1 if the array is empty or contains only string or object keys.

Otherwise, the first value is inserted at `Object.Length() + 1`, even if that position is negative or zero. If this is undesired and the object can contain negative keys, `Object.InsertAt(Object.Length() + 1, ...)` can be used instead.

# Pop

Removes and returns the last array element.

```
Value := Object.Pop()
```

If there are no array elements, the return value is an empty string. Otherwise, it is equivalent to the following:

```
Value := Object.RemoveAt(Object.Length())
```

# Delete

Removes key-value pairs from an object.

```
Object.Delete(Key)
Object.Delete(FirstKey, LastKey)
```

## Key

Any single key.

## FirstKey, LastKey

Any valid range of integer or string keys, where *FirstKey* <= *LastKey*. Both keys must be the same type.

## Return Value

If there is exactly one parameter, the removed value is returned (blank if none). Otherwise the return value is the number of matching keys which were found and removed.

## Remarks

Unlike [RemoveAt](#), Delete does not affect any of the key-value pairs that it does not remove. For example:

```
x := ["A", "B"]
MsgBox % x.RemoveAt(1) ; A
MsgBox % x[1] ; B
```

```
x := ["A", "B"]
MsgBox % x.Delete(1) ; A
MsgBox % x[1] ; Empty
```

## MinIndex / MaxIndex

```
MinIndex := Object.MinIndex()
```

```
MaxIndex := Object.MaxIndex()
```

If any integer keys are present, MinIndex returns the lowest and MaxIndex returns the highest. Otherwise an empty string is returned.

# Length

```
Length := Object.Length()
```

Returns the length of a linear array beginning at position 1; that is, the highest positive integer key contained by the object, or 0 if there aren't any.

```
MsgBox % ["A", "B", "C"].Length() ; 3
MsgBox % ["A", , "C"].Length() ; 3
MsgBox % {-10: 0, 10: 0}.Length() ; 10
MsgBox % {-10: 0, -1: 0}.Length() ; 0
```

## SetCapacity

Adjusts the capacity of an object or one of its fields.

```
Object.SetCapacity(MaxItems)
```

```
Object.SetCapacity(Key, ByteSize)
```

**MaxItems** The maximum number of key-value pairs the object should be able to contain before it must be automatically expanded. If less than the current number of key-value pairs, that number is used instead, and any unused space is freed.

**Key** Any valid key.

**ByteSize** The new size in bytes of the field's string buffer, excluding the null-terminator. If the field does not exist, it is created. If *ByteSize* is zero, the buffer is freed but the empty field is not removed. If *ByteSize* is less than the current size, excess data is truncated; otherwise all existing data is preserved.

**Returns** The new capacity if successful, otherwise an empty string.

## GetCapacity

```
MaxItems := Object.GetCapacity()
ByteSize := Object.GetCapacity(Key)
```

Returns the current capacity of an object or one of its fields.

## GetAddress

```
Ptr := Object.GetAddress(Key)
```

Returns the current address of the field's string buffer, if it has one.

## NewEnum

```
Enum := Object._NewEnum()
```

Returns a new [enumerator](#) to enumerate this object's key-value pairs. This method is usually not called directly, but by the [for-loop](#).

## HasKey

```
Object.HasKey(Key)
```

Returns true if *Key* is associated with a value (even "") within *Object*, otherwise false.

# Clone

```
Clone := Object.Clone([From, To])
```

**From** (optional) The lowest key (integer or string) to include in cloned object. When this parameter is missing, all strings and integers lower than **To** will be included.

**To** (optional) The highest key to include in cloned object. When this parameter is missing, all integers and strings higher than **From** will be included.

Note, the order of From integers To strings.

```
obj:=
{1:1,2:2,3:3,4:4,5:5,test:"test",ahk:"ahk",var:"var",(""): "empty",([]): "object"}
for k,v in obj.Clone(,obj.Length()) ; clone all integer keys
 out1.=k "=" v "`n"
MsgBox % out1
for k,v in obj.Clone("") ; clone all string keys
 out2.=k "=" v "`n"
MsgBox % out2
for k,v in obj.Clone("t","x") ; only string keys from t - x
 out3.=k "=" v "`n"
MsgBox % out3
for k,v in obj.Clone(2,4) ; only integer keys from 2,4
 out4.=k "=" v "`n"
MsgBox % out4
for k,v in obj.Clone() ; clone all
 out4.=k "=" v "`n"
```

MsgBox % out4

## ObjRawSet

Stores or overwrites a key-value pair in the object.

```
ObjRawSet Object, Key, Value
```

This function is provided to allow scripts to bypass the `__Set` meta-function.

If that isn't required, a normal assignment should be used instead. For

example: `Object[Key] := Value`

Since the purpose is to bypass meta-functions, this is a function only, not a method. Calling a built-in method generally causes the `__Call` meta-function to be called.

# Enumerator Object

Allows items in a collection to be enumerated.

## Next

Retrieves the next item or items in an enumeration.

```
Enum.Next(OutputVar1 [, OutputVar2, ...])
```

OutputVar1, OutputVar2    Receives an implementation-specific value.

...                      Additional parameters, if supported.

**Returns**                A non-zero integer if successful, zero if there were no items remaining, or an empty string if parameters were incorrect.

## Object

Enumerators returned by [ObjNewEnum](#) are called once for each key-value pair, and allow up to two parameters:

OutputVar1    Receives the **key** in a key-value pair.

OutputVar2    Receives the **value** associated with *OutputVar1*.

Key-value pairs are returned in an implementation-defined order. That is, they are typically not returned in the same order that they were assigned. Existing key-value pairs may be modified during enumeration, but inserting or removing keys may cause some items to be enumerated multiple times or not at all.

## Related

For-loop, Object.\_NewEnum

## Example

```
; Create some sample data.
```

```
obj := Object("red", 0xFF0000, "blue", 0x0000FF,
"green", 0x00FF00)
```

```
; Enumerate!
```

```
enum := obj._NewEnum()
while enum[k, v]
 t .= k "=" v "`n"
MsgBox % t
```

```
; or simply:
```

```
For k, v in obj
 s .= k "=" v "`n"
MsgBox % s
```

# File Object

Provides an interface for file input/output. `FileOpen` returns an object of this type.

# Read

Reads a string of characters from the file and advances the file pointer.

```
String := File.Read([Characters])
```

**Characters** The maximum number of characters to read. If omitted, the rest of the file is read and returned as one string. If the File object was created from a handle to a non-seeking device such as a console buffer or pipe, omitting this parameter may cause the method to fail or return only what data is currently available.

**Returns** A string.

## Write

Writes a string of characters to the file and advances the file pointer.

```
File.write(String)
```

String     A string.

**Returns**     The number of bytes (not characters) that were written.

## ReadLine

Reads a line of text from the file and advances the file pointer.

```
Line := File.ReadLine()
```

**Returns** A line of text, excluding the line ending.

Lines up to 65,534 characters long can be read. If the length of a line exceeds this, the remainder of the line is returned by subsequent calls to this method.

## WriteLine

Writes a string of characters followed by `\n` or `\r\n` depending on the flags used to open the file. Advances the file pointer.

```
File.WriteLine([String])
```

**String**      An optional string.

**Returns**    The number of bytes (not characters) that were written.

## ReadNum

Reads a number from the file and advances the file pointer.

```
Num := File.ReadNumType()
```

*NumType* One of the following specified directly as part of the function name:  
UInt, Int, Int64, Short, UShort, Char, UChar, Double, or Float.

These type names have the same meanings as with [DllCall](#).

**Returns** A number if successful, otherwise an empty string.

If a Try statement is active and no bytes were read, an exception is thrown. However, no exception is thrown if at least one byte was read, even if the size of the given *NumType* is greater than the number of bytes read. Instead, the missing bytes are assumed to be zero.

## WriteNum

Writes a number to the file and advances the file pointer.

```
File.WriteNumType(Num)
```

*NumType* One of the following specified directly as part of the function name:  
UInt, Int, Int64, Short, UShort, Char, UChar, Double, or Float.

These type names have the same meanings as with [DllCall](#).

Num A number.

**Returns** The number of bytes that were written. For instance, WriteUInt returns 4 if successful.

# RawRead

Read raw binary data from the file into memory.

```
File.RawRead(Buffer, Bytes)
```

**Buffer**     The variable or memory address which will receive the data.

**Bytes**     The maximum number of bytes to read.

**Returns**   The number of bytes that were read.

To read to a memory address, pass a pure integer such as `&variable + offset` or a variable containing a pure integer.

To read into a variable, pass a variable which is empty or contains a string. If *Bytes* is omitted, it defaults to the capacity of the variable. If *Bytes* is greater than the capacity of the variable, the variable is expanded. After the data is read, the variable's length is set to the string length of the data (rounded up to a whole number).

# RawWrite

Write raw binary data to the file.

```
File.RawWrite(Data, Bytes)
```

**Data**        A memory address or string containing binary data.

**Bytes**        The number of bytes to write.

**Returns**     The number of bytes that were written.

To write data from a given memory address, pass a pure integer such as `&variable + offset` or a variable containing a pure integer.

To write data from a string containing binary data, pass a string or a variable containing a string. If *Bytes* is omitted, it defaults to the string length times 2 (so is always an even number).

For an example, see [Saving and Restoring the Clipboard](#).

# Seek

Moves the file pointer.

```
File.Seek(Distance [, Origin := 0])
File.Position := Distance
File.Pos := Distance
```

**Distance** Distance to move, in bytes. Lower values are closer to the beginning of the file.

**Origin** Starting point for the file pointer move. Must be one of the following:

- 0 (SEEK\_SET): Beginning of the file. *Distance* must be zero or greater.
- 1 (SEEK\_CUR): Current position of the file pointer.
- 2 (SEEK\_END): End of the file. *Distance* should usually be negative.

If omitted, *Origin* defaults to SEEK\_END when *Distance* is negative and SEEK\_SET otherwise.

**Returns** A non-zero value if successful, otherwise zero.

## Tell

```
Pos := File.Tell()
Pos := File.Position
```

**Returns** The current position of the file pointer, where 0 is the beginning of the file.

## Length

Retrieves or sets the size of the file.

```
FileSize := File.Length
File.Length := NewSize
```

**NewSize** The new size of the file, in bytes.

**Returns** The size of the file, in bytes.

This property should be used only with an actual file. If the File object was created from a handle to a pipe, it may return the amount of data currently available in the pipe's internal buffer, but this behaviour is not guaranteed.

## AtEOF

```
IsAtEOF := File.AtEOF
```

**Returns** A non-zero value if the file pointer has reached the end of the file, otherwise zero.

This property should be used only with an actual file. If the File object was created from a handle to a non-seeking device such as a console buffer or pipe, the returned value may not be meaningful, as such devices do not logically have an "end of file".

## Close

Closes the file, flushes any data in the cache to disk and releases the share locks. Although the file is closed automatically when the object is freed, it is recommended to close the file as soon as possible.

```
File.Close()
```

*No parameters or return value.*

# Encoding

Retrieves or sets the text encoding used by this file object.

```
Encoding := File.Encoding
File.Encoding := Encoding
```

`Encoding` A numeric code page identifier (see [MSDN](#)) or one of the following strings:

- `UTF-8`: Unicode UTF-8, equivalent to CP65001.
- `UTF-16`: Unicode UTF-16 with little endian byte order, equivalent to CP1200.
- `CP $nnn$` : a code page with numeric identifier  $nnn$ .

`Encoding` never returns a value with the `-RAW` suffix, regardless of how the file was opened or whether it contains a byte order mark (BOM). Setting `Encoding` never causes a BOM to be added or removed, as the BOM is normally written to the file when it is first created.

Setting `Encoding` to `UTF-8-RAW` or `UTF-16-RAW` is valid in v1.1.15.04+, but the `-RAW` suffix is ignored. In earlier versions, `UTF-8-RAW` and `UTF-16-RAW` behaved like an invalid 8-bit encoding, causing all non-ASCII characters to be discarded. This only applies to `File.Encoding`, not `FileOpen()`.

## **\_\_Handle**

`File.__Handle`

**Returns** A system file handle, intended for use with `DllCall`. See [CreateFile](#).

File objects internally buffer reads or writes. If data has been written into the object's internal buffer, it is committed to disk before the handle is returned. If the buffer contains data read from file, it is discarded and the actual file pointer is reset to the logical position indicated by `File.Pos`.

# Function Objects

"Function object" usually means any of the following:

- A reference to a [Func object](#), which represents an actual [function](#); either built-in or defined by the script.
- A user-defined object which can be called like a function. This is sometimes also referred to as a "functor".
- Any other object which can be called like a function, such as a [BoundFunc object](#) or a JavaScript function object returned by a COM method.

Function objects can be used with the following:

- [Gui events](#)
- [Hotkey](#)
- [Menu](#)
- [OnClipboardChange\(\)](#)
- [OnExit\(\)](#)
- [OnMessage\(\)](#)
- [SetTimer](#)

## User-Defined

User-defined function objects must define a *Call* method containing the implementation of the "function".

```
class YourClassName {
 Call(a, b) { ; Declare parameters as
needed, or an array".
 ;...
 return c
 }
 ;...
}
```

## Examples

The following example defines a function array which can be called; when called, it calls each element of the array in turn.

```
class FuncArrayType {
 Call(obj, params*) {
 ; Call a list of functions.
 Loop % this.Length()
 this[A_Index].Call(params*)
 }
}

; Create an array of functions.
funcArray := new FuncArrayType
; Add some functions to the array (can be done
at any point).
funcArray.Push(Func("One"))
```

```
funcArray.Push(Func("Two"))
; Create an object which uses the array as a
method.
```

```
obj := {method: funcArray}
```

```
; Call the method.
```

```
obj.method("foo", "bar")
```

```
One(param1, param2) {
```

```
 ListVars
```

```
 MsgBox
```

```
}
```

```
Two(param1, param2) {
```

```
 ListVars
```

```
 MsgBox
```

```
}
```

## BoundFunc Object

Acts like a function, but just passes predefined parameters to another function.

There are two ways that BoundFunc objects can be created:

- By calling the `Func.Bind()` method, which binds parameter values to a function.
- By calling the `ObjBindMethod()` function, which binds parameter values and a method name to a target object.

BoundFunc objects can be called as shown in the example below. No other methods are supported. When the BoundFunc is called, it calls the function or method to which it is bound, passing any bound parameters followed by any which were passed by the caller. For example:

```
fn := Func("RealFn").Bind(1)

%fn%(2) ; Shows "1, 2"
fn.Call(3) ; Shows "1, 3"

RealFn(a, b) {
 MsgBox("%a%, %b%")
}
```

`ObjBindMethod()` can be used to bind to a method when it isn't possible to retrieve a reference to the method itself. For example:

```
file := FileOpen(A_ScriptFullPath, "r")
```

```
getLine := ObjBindMethod(file, "ReadLine")
MsgBox % %getLine%() ; Shows the first line of
this file.
```

For a more complex example, see [SetTimer](#).

# Frequently Asked Questions (FAQ)

If your question is not in the list below, check the community-driven [FAQ](#) on [autoHotkey.net](#).

## Language Syntax

- When are quotation marks used with commands and their parameters?
- When exactly are variable names enclosed in percent signs?
- When should percent signs and commas be escaped?

## General Troubleshooting

- What can I do if AutoHotkey won't install?
- Why do some lines in my script never execute?
- Why doesn't my script work on Windows *xxx* even though it worked on a previous version?
- How do I work around problems caused by User Account Control (UAC)?
- I can't edit my script via tray icon because it won't start due to an error. Can I find my script somewhere else?
- How can I find and fix errors in my code?
- Why is the Run command unable to launch my game or program?
- Why don't Hotstrings, Send, and MouseClick work in certain games?
- How can performance be improved for games or at other times when the CPU is under heavy load?
- My antivirus program flagged AHK as malware. Does it really contain a

virus?

## Common Tasks

- Where can I find the official build, or older releases?
- Can I run AHK from a USB drive?
- How can the output of a command line operation be retrieved?
- How can a script close, pause, or suspend other script(s)?
- How can a repeating action be stopped without exiting the script?
- How can context sensitive help for AutoHotkey commands be used in any editor?
- How to detect when a web page is finished loading?
- How can dates and times be compared or manipulated?
- How can I send the current Date and/or Time?
- How can I send text to a window which isn't active or isn't visible?
- How can Winamp be controlled even when it isn't active?
- How can MsgBox's button names be changed?
- How can I change the default editor, which is accessible via context menu or tray icon?
- How can I save the contents of my GUI controls?
- Can I draw something with AHK?
- How can I start an action when a window appears, closes or becomes [in]active?

## Hotkeys, Hotstrings, and Remapping

- How do I put my hotkeys and hotstrings into effect automatically every time I start my PC?
- I'm having trouble getting my mouse buttons working as hotkeys. Any advice?
- How can tab and space be defined as hotkeys?
- How can keys or mouse buttons be remapped so that they become different keys?
- How do I detect the double press of a key or button?
- How can a hotkey or hotstring be made exclusive to certain program(s)? In other words, I want a certain key to act as it normally does except when a specific window is active.
- How can a prefix key be made to perform its native function rather than doing nothing?
- How can the built-in Windows shortcut keys, such as Win+U (Utility Manager) and Win+R (Run), be changed or disabled?
- Can I use wildcards or regular expressions in Hotstrings?
- How can I use a hotkey that is not in my keyboard layout?
- My keypad has a special 000 key. Is it possible to turn it into a hotkey?

## Language Syntax

### When are quotation marks used with commands and their parameters?

Double quotes (") have special meaning only within [expressions](#). In all other places, they are treated literally as if they were normal characters. However, when a script launches a program or document, the operating system usually requires quotes around any command-line parameter that contains spaces, such as in this example: `Run, Notepad.exe "C:\My Documents\Address List.txt"`.

### When exactly are variable names enclosed in percent signs?

1) Variable names and other expressions inside strings (text) are always enclosed in percent signs. For example:

```
MsgBox, Hello`, %A_UserName%!
MsgBox("Have a nice day, %A_UserName%!")
```

2) Variable names may also be enclosed in percent signs to perform a [double-deref](#). In such cases, the variable must contain the name of another variable.

Aside from double-derefs and strings, percent signs are not used in the cases illustrated in **bold** below:

1) In parameters that are output variables: `StrLen, OutputVar`

```
%InputVar%
```

2) Everywhere else in [expressions](#). For example:

```
If (Var1 <> Var2)
 Var1 := Var2 + 100
MsgBox("The value is " Var1)
```

## When should percent signs and commas be escaped?

Literal percent signs must be [escaped](#) by preceding them with an accent/backtick. For example:

```
MsgBox The current percentage is 25`%.
MsgBox("The current percentage is 25`%.")
```

Literal commas must also be escaped ( ``,`` ) except when used in `MsgBox` (in which case the accent is permitted but not necessary).

When commas are enclosed in quotes within an [expression](#), the accent is permitted but not necessary. For example: `MsgBox("Hello, world!")`.

## General Troubleshooting

### What can I do if AutoHotkey won't install?

**7-zip Error:** Use 7-zip or a compatible program to extract the setup files from the installer EXE, then run setup.exe or Installer.ahk (drag and drop Installer.ahk onto AutoHotkeyU32.exe).

AutoHotkey's installer comes packaged as a 7-zip self-extracting archive which attempts to extract to the user's Temp directory and execute a compiled script. Sometimes system policies or other factors prevent the files from being extracted or executed. Usually in such cases the message "7-zip Error" is displayed. Manually extracting the files to a different directory may help.

**Setup hangs:** If the setup window comes up blank or not at all, try one or both of the following:

- Hold Ctrl or Shift when the installer starts. If you get a UAC prompt, hold Ctrl or Shift as you click Yes/Continue. You should get a prompt asking whether you want to install with default options.
- Install using [command line options](#). If you have manually extracted the setup files from the installer EXE, use either `setup.exe /S` or `AutoHotkeyU32.exe Installer.ahk /S`.

**Other:** The suggestions above cover the most common problems. For further assistance, post on the forums.

## Why do some lines in my script never execute?

Any lines you want to execute immediately when the script starts should appear at the top of the script, prior to the first [hotkey](#), [hotstring](#), or [Return](#). For details, see [auto-execute section](#).

Also, a [hotkey](#) that executes more than one line must list its first line *beneath* the hotkey, not on the same line. For example:

```
#space:: ; Win+Spacebar
Run Notepad
WinWaitActive Untitled - Notepad
WinMaximize
return
```

## Why doesn't my script work on Windows xxx even though it worked on a previous version?

There are many variations of this problem, such as:

- I've upgraded my computer/Windows and now my script won't work.
- Hotkeys/hotstrings don't work when a program running as admin is active.
- Some windows refuse to be automated (e.g. Device Manager ignores Send).

If you've switched operating systems, it is likely that something else has also changed and may be affecting your script. For instance, if you've got a new computer, it might have different drivers or other software installed. If you've also updated to a newer version of AutoHotkey, find out which version you had

before and then check the [changelog](#) and [compatibility notes](#).

[SoundGet](#) and [SoundSet](#) behave differently on Vista and later than on earlier versions of Windows. In particular, device numbers are different and some components may be unavailable. Behaviour depends on the audio drivers, which are necessarily different to the ones used on XP. The [soundcard analysis script](#) can be used to find the correct device numbers.

Also refer to the following question:

## **How do I work around problems caused by User Account Control (UAC)?**

[User Account Control \(UAC\)](#) is a common cause of problems, especially when moving from Windows XP/Vista/7 to Vista/7/8/10. Although it is present in Windows Vista and later, it is enabled by default on new systems or new installs, and it is more difficult to disable on Windows 8 and later.

By default, UAC protects "elevated" programs (that is, programs which are running as admin) from being automated by non-elevated programs, since that would allow them to bypass security restrictions. Hotkeys are also blocked, so for instance, a non-elevated program cannot spy on input intended for an elevated program.

UAC may also prevent [SendPlay](#) and [BlockInput](#) from working.

Common workarounds are as follows:

- Enable the *Add 'Run with UI Access' to context menus* option in AutoHotkey Setup. This option can be enabled or disabled without reinstalling AutoHotkey by re-running AutoHotkey Setup from the Start menu. Once it is enabled, launch your script file by right-clicking it and selecting *Run with UI Access*, or use a *command line* like `"AutoHotkeyU32_UIA.exe" "Your script.ahk"` (but include full paths).
- Run the script *as administrator*. Note that this also causes any programs launched by the script to run as administrator, and may require the user to accept an approval prompt when launching the script.
- Disable the local security policy "Run all administrators in Admin Approval Mode" (not recommended).
- Disable UAC completely. This is not recommended, and is not feasible on Windows 8 or later.

## **I can't edit my script via tray icon because it won't start due to an error. What do I do?**

You need to fix the error in your script before you can get your tray icon back. But first, you need to find the script file.

Look for AutoHotkey.ahk in the following directories:

- Your *Documents* (or *My Documents*) folder.
- The directory where you installed AutoHotkey, usually C:\Program Files\AutoHotkey. If you are using AutoHotkey without having installed it,

look in the directory which contains AutoHotkey.exe.

If you are running another AutoHotkey executable directly, the name of the script depends on the executable. For example, if you are running AutoHotkeyU32.exe, look for AutoHotkeyU32.ahk. Note that depending on your system settings the ".ahk" part may be hidden, but the file should have an icon like 

You can usually edit a script file by right clicking it and selecting *Edit Script*. If that doesn't work, you can open the file in Notepad or another editor.

If you launch AutoHotkey from the Start menu or by running AutoHotkey.exe directly (without command line parameters), it will look for a script in one of the locations shown above. Alternatively, you can create a script file (something.ahk) anywhere you like, and run the script file instead of running AutoHotkey.

See also [Command Line Parameter "Script Filename"](#) and [Portability of AutoHotkey.exe](#).

## How can I find and fix errors in my code?

For simple scripts, see [Debugging a Script](#). To show contents of a variable, use [MsgBox](#) or [ToolTip](#). For complex scripts, see [Interactive Debugging](#).

## Why is the Run command unable to launch my game or program?

Some programs need to be started in their own directories (when in doubt, it is

usually best to do so). For example:

```
Run, %A_ProgramFiles%\Some Application\App.exe,
%A_ProgramFiles%\Some Application
```

If the program you are trying to start is in `%A_winDir%\System32` and you are using AutoHotkey 32-bit on a 64-bit system, the [File System Redirector](#) may be interfering. To work around this, use `%A_winDir%\SysNative` instead; this is a virtual directory only visible to 32-bit programs running on 64-bit systems.

## Why do Hotstrings, Send, and Click have no effect in certain games?

Not all games allow AHK to send keys and clicks or receive pixel colors.

But there are some alternatives, try all the solutions mentioned below. If all these fail, it may not be possible for AHK to work with your game. Sometimes games have a hack and cheat prevention measure, such as GameGuard and Hackshield. If they do, there is a high chance that AutoHotkey will not work with that game.

- Use SendPlay via the [SendPlay](#) command, [SendMode Play](#) and/or the [hotstring option SP](#).

```
SendPlay, abc
```

```
SendMode, Play
Send, abc
```

```
:SP:btw::by the way
```

```
; or
```

```
#Hotstring SP
::btw::by the way
```

Note: SendPlay may have no effect at all on Windows Vista or later if User Account Control is enabled, even if the script is running as an administrator.

- Increase [SetKeyDelay](#). For example:

```
SetKeyDelay, 0, 50
SetKeyDelay, 0, 50, Play
```

- Try [ControlSend](#), which might work in cases where the other Send modes fail:

```
ControlSend,, abc, game_title
```

- Try the down and up event of a key with the various send methods:

```
Send {KEY down}{KEY up}
```

- Try the down and up event of a key with a [Sleep](#) between them:

```
Send {KEY down}
Sleep 10 ; try various milliseconds
Send {KEY Up}
```

## How can performance be improved for games or at other times when the CPU is under heavy load?

If a script's [Hotkeys](#), [Clicks](#), or [Sends](#) are noticeably slower than normal while the CPU is under heavy load, raising the script's process-priority may help. To do this, include the following line near the top of the script:

```
ProcessSetPriority High
```

## My antivirus program flagged AutoHotkey or a compiled script as malware. Is it really a virus?

Although it is certainly possible that the file has been infected, most often these alerts are *false positives*, meaning that the antivirus program is mistaken. One common suggestion is to upload the file to an online service such as [virustotal](#) or [Jotti](#) and see what other antivirus programs have to say. If in doubt, you could send the file to the vendor of your antivirus software for confirmation. This might also help us and other AutoHotkey users, as the vendor may confirm it is a false positive and fix their product to play nice with AutoHotkey.

False positives might be more common for compiled scripts which have been compressed, such as with UPX (default for AutoHotkey 1.0 but not 1.1) or MPRESS (optional for AutoHotkey 1.1). As the default AutoHotkey installation does not include a compressor, compiled scripts are not compressed by default.

## Common Tasks

### Where can I find the official build, or older releases?

See [download page of AutoHotkey](#).

### Can I run AHK from a USB drive?

See [Portability of AutoHotkey.exe](#). Note that if you use auto-included function libraries, AutoHotkey.exe and the Lib folder must be up one level from Ahk2Exe.exe (e.g. \AutoHotkey.exe vs \Compiler\Ahk2Exe.exe). Also note that Ahk2Exe saves settings to the following registry key:

```
HKEY_CURRENT_USER\Software\AutoHotkey\Ahk2Exe.
```

### How can the output of a command line operation be retrieved?

Testing shows that due to file caching, a temporary file can be very fast for relatively small outputs. In fact, if the file is deleted immediately after use, it often does not actually get written to disk. For example:

```
RunWait %comspec% /c dir > C:\My Temp File.txt
FileRead, VarToContainContents, C:\My Temp
File.txt
FileDelete, C:\My Temp File.txt
```

To avoid using a temporary file (especially if the output is large), consider using the [Shell.Exec\(\)](#) method as shown in the examples for the [Run](#) command.

## How can a script close, pause, or suspend other script(s)?

First, here is an example that closes another script:

```
DetectHiddenWindows On ; Allows a script's hidden main window to be detected.
SetTitleMatchMode 2 ; Avoids the need to specify the full path of the file below.
WinClose Script's File Name.ahk - AutoHotkey ; Update this to reflect the script's name (case sensitive).
```

To suspend or pause another script, replace the last line above with one of these:

```
PostMessage, 0x111, 65305,,, Script's File Name.ahk - AutoHotkey ; Suspend.
PostMessage, 0x111, 65306,,, Script's File Name.ahk - AutoHotkey ; Pause.
```

## How can a repeating action be stopped without exiting the script?

To pause or resume the entire script at the press of a key, assign a hotkey to the `Pause` command as in this example:

```
^!p::Pause ; Press Ctrl+Alt+P to pause. Press it again to resume.
```

To stop an action that is repeating inside a `Loop`, consider the following working example, which is a hotkey that both starts and stops its own repeating action. In other words, pressing the hotkey once will start the `Loop`. Pressing the same

hotkey again will stop it.

```
#MaxThreadsPerHotkey 3
#Z:: ; Win+Z hotkey (change this hotkey to
suit your preferences).
#MaxThreadsPerHotkey 1
if KeepWinZRunning ; This means an underlying
thread is already running the loop below.
{
 KeepWinZRunning := false ; Signal that
thread's loop to stop.
 return ; End this thread so that the one
underneath will resume and see the change made
by the line above.
}
; Otherwise:
KeepWinZRunning := true
Loop
{
 ; The next four lines are the action you
want to repeat (update them to suit your
preferences):
 ToolTip, Press Win-Z again to stop this
from flashing.
 Sleep 1000
 ToolTip
 Sleep 1000
 ; But leave the rest below unchanged.
 if not KeepWinZRunning ; The user signaled
the loop to stop by pressing Win-Z again.
 break ; Break out of this loop.
}
KeepWinZRunning := false ; Reset in
preparation for the next press of this hotkey.
return
```

## How can context sensitive help for AutoHotkey commands be used in any editor?

Rajat created [this script](#).

## How to detect when a web page is finished loading?

With Internet Explorer, perhaps the most reliable method is to use DllCall and COM as demonstrated at [www.autohotkey.com/forum/topic19256.html](http://www.autohotkey.com/forum/topic19256.html). On a related note, the contents of the address bar and status bar can be retrieved as demonstrated at [www.autohotkey.com/forum/topic19255.html](http://www.autohotkey.com/forum/topic19255.html).

**Older, less reliable method:** The technique in the following example will work with MS Internet Explorer for most pages. A similar technique might work in other browsers:

```
Run, www.yahoo.com
MouseMove, 0, 0 ; Prevents the status bar from
showing a mouse-hover link instead of "Done".
WinWait, Yahoo! -
WinActivate
StatusBarWait, Done, 30
if ErrorLevel
 MsgBox The wait timed out or the window was
closed.
else
 MsgBox The page is done loading.
```

## How can dates and times be compared or manipulated?

The `DateAdd` command can add or subtract a quantity of days, hours, minutes, or seconds to a time-string that is in the `YYYYMMDDHH24MISS` format. The following example subtracts 7 days from the specified time:

```
Result := DateAdd(VarContainingTimestamp, -7, "days")
```

To determine the amount of time between two dates or times, see `DateDiff`, which gives an example. Also, the built-in variable `A_Now` contains the current local time. Finally, there are several built-in [date/time variables](#), as well as the `FormatTime` command to create a custom date/time string.

## How can I send the current Date and/or Time?

Use `FormatTime` or [built-in variables for date and time](#).

## How can I send text to a window which isn't active or isn't visible?

Use `ControlSend`.

## How can Winamp be controlled even when it isn't active?

See [Automating Winamp](#).

## How can MsgBox's button names be changed?

Here is an [example](#).

## How can I change the default editor, which is accessible via context menu or tray icon?

In the example section of [Edit](#) you will find a script that allows you to change the default editor.

## How can I save the contents of my GUI controls?

Use [Gui.Submit](#). For Example:

```
Gui := GuiCreate()
Gui.Add("Text",, "Enter some Text and press
Submit:")
Gui.Add("Edit", "vMyEdit")
Gui.Add("Button",, "Submit").OnEvent("Click",
"Submit")
Gui.Show()

Submit(Btn)
{
 Saved := Btn.Gui.Submit(false)
 MsgBox("Content of the edit control: "
Saved.MyEdit)
}
```

## Can I draw something with AHK?

See [GDI+ standard library](#) by tic. It's also possible with some rudimentary methods using Gui, but in a limited way.

## How can I start an action when a window appears, closes or

## becomes [in]active?

Use `WinWait`, `WinWaitClose` or `WinWait[Not]Active`. See the [community-driven FAQ](#) (windows section) for more possibilities.

## Hotkeys, Hotstrings, and Remapping

### How do I put my hotkeys and hotstrings into effect automatically every time I start my PC?

There is a folder in the Start Menu called Startup. If you put a shortcut to your script in that folder, the script will launch automatically every time you start your PC. To create a shortcut:

1. Find the script file, select it, and press Control-C.
2. Right-click the Start button (typically at the lower left corner of the screen) and choose "Explore All Users".
3. Navigate to the Startup folder inside the Programs folder.
4. From the menu bar, choose *Edit -> Paste Shortcut*. The shortcut to the script should now be in the Startup folder.

### I'm having trouble getting my mouse buttons working as hotkeys. Any advice?

The left and right mouse buttons should be assignable normally (for example, `#LButton::` is the Win+LeftButton hotkey). Similarly, the middle button and the turning of the [mouse wheel](#) should be assignable normally except on mice whose drivers directly control those buttons.

The fourth button (XButton1) and the fifth button (XButton2) might be assignable if your mouse driver allows their clicks to be [seen](#) by the system. If

they cannot be seen -- or if your mouse has more than five buttons that you want to use -- you can try configuring the software that came with the mouse (sometimes accessible in the Control Panel or Start Menu) to send a keystroke whenever you press one of these buttons. Such a keystroke can then be defined as a hotkey in a script. For example, if you configure the fourth button to send Control+F1, you can then indirectly configure that button as a hotkey by using `^F1:::` in a script.

If you have a five-button mouse whose fourth and fifth buttons cannot be seen, you can try changing your mouse driver to the default driver included with the OS. This assumes there is such a driver for your particular mouse and that you can live without the features provided by your mouse's custom software.

## How can Tab and Space be defined as hotkeys?

Use the names of the keys (Tab and Space) rather than their characters. For example, `#Space` is Win+Space and `^!Tab` is Control+Alt+Tab.

## How can keys or mouse buttons be remapped so that they become different keys?

This is described on the [remapping](#) page.

## How do I detect the double press of a key or button?

Use [built-in variables for hotkeys](#) as follows:

```
~Ctrl::
 if (A_ThisHotkey = A_PriorHotkey &&
A_TimeSincePriorHotkey < 200)
 MsgBox double-press
return
```

**How can a hotkey or hotstring be made exclusive to certain program(s)? In other words, I want a certain key to act as it normally does except when a specific window is active.**

The preferred method is `#IfWinActive`. For example:

```
#IfWinActive, ahk_class Notepad
^a::MsgBox You pressed Control-A while Notepad
is active.
```

**How can a prefix key be made to perform its native function rather than doing nothing?**

Consider the following example, which makes Numpad0 into a prefix key:

```
Numpad0 & Numpad1::MsgBox, You pressed Numpad1
while holding down Numpad0.
```

Now, to make Numpad0 send a real Numpad0 keystroke whenever it wasn't used to launch a hotkey such as the above, add the following hotkey:

```
$Numpad0::Send, {Numpad0}
```

The \$ prefix is needed to prevent a warning dialog about an infinite loop (since the hotkey "sends itself"). In addition, the above action occurs at the time the key is released.

## **How can the built-in Windows shortcut keys, such as Win+U (Utility Manager) and Win+R (Run), be changed or disabled?**

Here are some [examples](#).

## **Can I use wildcards or regular expressions in Hotstrings?**

Use the [script](#) by polyethene (examples are included).

## **How can I use a hotkey that is not in my keyboard layout?**

See [Special Keys](#).

## **My keypad has a special 000 key. Is it possible to turn it into a hotkey?**

You can. This [example script](#) makes the 000 key into an equals key. You can change the action by replacing the "Send, =" line with line(s) of your choice.

# AutoHotkey\_L New Features

[\[home\]](#)

| Control Flow                                              |                                                                                                                                                                                                                    |
|-----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Break</a> <i>LoopLabel</i>                    | Break out of a loop or any number of nested loops.                                                                                                                                                                 |
| <a href="#">Continue</a> <i>LoopLabel</i>                 | Continue a loop, even from within any number of nested loops.                                                                                                                                                      |
| <a href="#">For</a> <i>x y in z</i>                       | Loop through the contents of an object.                                                                                                                                                                            |
| <a href="#">Loop Until</a>                                | Loop until a condition is true. Applicable to any type of Loop.                                                                                                                                                    |
| <a href="#">Try...Catch</a>                               | Provides structured exception handling.                                                                                                                                                                            |
| <a href="#">Throw</a>                                     | Throws an exception.                                                                                                                                                                                               |
| Commands                                                  |                                                                                                                                                                                                                    |
| <a href="#">FileEncoding</a>                              | Sets the default encoding for <a href="#">FileRead</a> , <a href="#">FileReadLine</a> , <a href="#">Loop Read</a> , <a href="#">FileAppend</a> , and <a href="#">FileOpen</a> .<br><i>See also: Text Encodings</i> |
| <a href="#">Gui</a>                                       | See <a href="#">GUI Enhancements</a> below.                                                                                                                                                                        |
| <a href="#">IniRead/Write/Delete</a>                      | Read, write or delete entire sections, or retrieve a list of all section names.                                                                                                                                    |
| <a href="#">Menu, Icon</a>                                | Sets or removes a menu item's icon.                                                                                                                                                                                |
| <a href="#">Run</a>                                       | <a href="#">Improvements</a> were made to the way parameters are parsed.                                                                                                                                           |
| <a href="#">SendInput</a> {U+nnnn}                        | Sends a Unicode character. Unicode characters may be used directly in Unicode builds.                                                                                                                              |
| <a href="#">SendLevel</a>                                 | Controls which artificial keyboard and mouse events are ignored by hotkeys and hotstrings.                                                                                                                         |
| <a href="#">SetFormat</a> , <a href="#">IntegerFast</a> , |                                                                                                                                                                                                                    |

|                                                                                                                                                       |                                                                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>h h</code>                                                                                                                                      | Set lower-case or upper-case hexadecimal format.                                                                                                                  |
| <code>Transform, HTML</code>                                                                                                                          | Perform code page or HTML transformations.                                                                                                                        |
| <code>WinGet, ..., ProcessPath</code>                                                                                                                 | Retrieves the full path and name of the process that owns a given window.                                                                                         |
| <b>Directives</b>                                                                                                                                     |                                                                                                                                                                   |
| <code>#If <i>expression</i></code>                                                                                                                    | Similar to <code>#IfWinActive</code> , but for arbitrary expressions.                                                                                             |
| <code>#IfTimeout</code>                                                                                                                               | Sets the maximum time that may be spent evaluating a single <code>#If</code> expression.                                                                          |
| <code>#MenuMaskKey</code>                                                                                                                             | Changes which key is used to mask Win or Alt keyup events.                                                                                                        |
| <code>#Include &lt;Lib&gt;</code>                                                                                                                     | Includes a script file from a function library folder.                                                                                                            |
| <code>#InputLevel</code>                                                                                                                              | Controls which artificial keyboard and mouse events are ignored by hotkeys and hotstrings.                                                                        |
| <code>#Warn</code>                                                                                                                                    | Enables or disables warnings for selected conditions that may be indicative of developer errors.                                                                  |
| <b>Functions</b>                                                                                                                                      |                                                                                                                                                                   |
| <code>ComObj...</code> --<br><code>ComObjActive</code><br><code>ComObjEnwrap/Unwrap</code><br><code>ComObjParameter</code><br><code>ComObjType</code> | Retrieves a registered COM object.<br>Wraps/unwraps a COM object.<br>Wraps a value and type to pass as a parameter.<br>Retrieves a COM object's type information. |
| <code>ComObjArray</code>                                                                                                                              | Creates a SAFEARRAY for use with COM.                                                                                                                             |
| <code>ComObjConnect</code>                                                                                                                            | Connects a COM object's event sources to functions with a given prefix.                                                                                           |
| <code>ComObjCreate</code>                                                                                                                             | Creates a COM object.                                                                                                                                             |
| <code>ComObjError</code>                                                                                                                              | Enables or disables notification of COM errors.                                                                                                                   |

|                             |                                                                                                      |
|-----------------------------|------------------------------------------------------------------------------------------------------|
| ComObjFlags                 | Retrieves or changes flags which control a COM wrapper object's behaviour.                           |
| ComObjGet                   | Returns a reference to an object provided by a COM component.                                        |
| ComObjQuery                 | Queries a COM object for an interface or service.                                                    |
| ComObjType                  | Retrieves type information from a COM object.                                                        |
| ComObjValue                 | Retrieves the value or pointer stored in a COM wrapper object.                                       |
| Exception                   | Creates an exception object for <code>Throw</code> (also provides limited access to the call stack). |
| FileOpen                    | Provides object-oriented file I/O.                                                                   |
| Func                        | Retrieves a <code>reference</code> to a function.                                                    |
| GetKeyName/VK/SC            | Retrieves the name or text, virtual key code or scan code of a key.                                  |
| InStr                       | Searches for a given <i>occurrence</i> of a string, from the left or the right.                      |
| IsByRef                     | Determines whether a ByRef parameter was supplied with a variable.                                   |
| IsObject                    | Determines whether a value is an object.                                                             |
| StrPut                      | Copies a string to a memory address, optional converting it between code pages.                      |
| StrGet                      | Reads a string from a memory address, optional converting it between code pages.                     |
| Trim                        | Trims certain characters from the beginning and/or end of a string.                                  |
| RegEx ( <i>?CNum Func</i> ) | Calls a function during evaluation of a regex pattern.                                               |
| Function Libraries          | New "local library" and <code>#Include &lt;LibName&gt;</code> .                                      |
|                             |                                                                                                      |

|                     |                                                                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Variadic Functions  | Functions may accept a variable number of parameters via an array.                                                              |
| Static Initializers | Static variables can now be initialized using any expression.                                                                   |
| Objects             |                                                                                                                                 |
| General             | Behaviour and usage of objects in general.                                                                                      |
| Object              | Associative arrays which can be extended with other functionality.                                                              |
| Enumerator          | Allows items in a collection to be enumerated.                                                                                  |
| File                | Provides an interface to access a file. <a href="#">FileOpen</a> returns an object of this type.                                |
| Func                | Represents a user-defined or built-in function which can be called by the script.                                               |
| ComObject           | See ComObj functions above.                                                                                                     |
| Variables           |                                                                                                                                 |
| A_IsUnicode         | In Unicode builds, this variable contains 1 ( <i>true</i> ). In ANSI builds it is not defined, so is effectively <i>false</i> . |
| A_FileEncoding      | Contains the default encoding for various commands; see <a href="#">FileEncoding</a> .                                          |
| A_OSVersion         | Contains WIN_7 on Windows 7, else one of the values documented <a href="#">here</a> .                                           |
| A_PriorKey          | The name of the last key which was pressed prior to the most recent key-press or key-release ... <a href="#">(More)</a>         |
| A_PtrSize           | Contains the size of a pointer, in bytes. This is either 4 (32-bit) or 8 (64-bit).                                              |
| A_ScriptHwnd        | The unique ID (HWND/handle) of the script's hidden main window.                                                                 |
| Datatypes           |                                                                                                                                 |

|                  |                                                                                                                                                                         |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ptr              | Equivalent to <i>Int</i> in 32-bit builds and <i>Int64</i> in 64-bit builds. Supported by <a href="#">DllCall</a> , <a href="#">NumPut</a> and <a href="#">NumGet</a> . |
| AStr, WStr       | Supported only by <a href="#">DllCall</a> ; see <a href="#">Script Compatibility</a> .                                                                                  |
| Unicode          |                                                                                                                                                                         |
| Compatibility    | How to deal with Unicode in <a href="#">DllCall</a> , etc.                                                                                                              |
| Script Files     | Using Unicode in script files.                                                                                                                                          |
| SendInput        | Using Unicode with <a href="#">SendInput</a> .                                                                                                                          |
| Other            |                                                                                                                                                                         |
| ahk_exe          | Windows can be identified by the name or path of the process (EXE file) which owns them.                                                                                |
| Debugging        | Interactive debugging features (line by line execution etc.).                                                                                                           |
| Error Handling   | Try/catch/throw and increased usefulness for <a href="#">A_LastError</a> .                                                                                              |
| GUI Enhancements | Various enhancements to the <a href="#">Gui</a> command and related.                                                                                                    |
| Icon Support     | Resource identifiers and improved support for various icon sizes.                                                                                                       |
| Other Changes    | Changes affecting script compatibility.                                                                                                                                 |
| Version History  | History of <a href="#">AutoHotkey_L</a> revisions.                                                                                                                      |
| Version History  | History of <a href="#">AutoHotkey_H</a> revisions.                                                                                                                      |

## Error Handling

Many commands support using `try/catch` instead of `ErrorLevel` for error handling. For example:

```
try
{
 FileCopy, file1.txt, C:\folder
 FileDelete, C:\folder\old.txt
}
catch
 MsgBox An error occurred!
```

Additionally, the following commands now set `A_LastError` to assist with debugging: `FileAppend`, `FileRead`, `FileReadLine`, `FileDelete`, `FileCopy`, `FileMove`, `FileGetAttrib/Time/Size/Version`, `FileSetAttrib/Time`, `FileCreateDir`, `RegRead`, `RegWrite`, `RegDelete`.

## Function Libraries

In addition to the user library in `%A_MyDocuments%\AutoHotkey\Lib` and standard library in the AutoHotkey directory, functions may be auto-included from the "local library" which resides in `%A_ScriptDir%\Lib`. For more information, see [Libraries of Functions](#).

`#Include <LibName>` explicitly includes a library file which may be located in any one of the function libraries.

## GUI Enhancements

A number of enhancements have been made to the `Gui` command and related:

- A `name` or `HWND` can be used instead of a number between 1 and 99 when referring to a GUI.
- `Gui, New` creates a new anonymous GUI.
- Any number of named or anonymous GUIs can be created.
- New GUI options: `+HwndOutputVar`, `+ParentGUI`
- A GUI's owner can be any arbitrary window: `+Owner%HWND%`.
- `Gui, Font` can control anti-aliasing of text.
- `ActiveX` controls such as the Internet Explorer WebBrowser control are supported.
- `GuiControlGet, OutputVar, Name` gets the name of the variable associated with a GUI control.
- Keyboard accelerators such as `Ctrl+O` are supported automatically when used in `Gui` menus.
- `Font quality` can be controlled by the `Font` sub-command.

## Static Variables

Static variables can now be initialized using any expression. For example:

```
Sleep 500
MsgBox % Time() "ms since the script started."
Time() {
 static Tick := A_TickCount
 return A_TickCount - Tick
}
```

## Text Encodings

`FileRead`, `FileReadLine`, `Loop Read` and `FileAppend` support the majority of Windows-supported text encodings, not just the system default ANSI code page. `FileEncoding` can be used to set the default encoding, which can be overridden for `FileRead` and `FileAppend` as follows:

```
OutPutVar := FileRead("*Pnnn Filename")
FileAppend [Text, Filename, Encoding]
```

While `nnn` must be a numeric *code page identifier*, *Encoding* follows the same format as `FileEncoding`.

**See also:** [Script Compatibility](#)

## Variadic Functions and Function-Calls

Variadic functions can receive a variable number of parameters via an array, while variadic function-calls can be used to pass a variable number of parameters to a function.

## Improvements to Icon Support

### Unusual Sizes

Icon resources of any size supported by the operating system may be extracted from executable files. When multiple sized icon resources exist within an icon group, the most appropriate size is used. Prior to revision 17, an arbitrary icon resource was selected by the system, scaled to the system large icon size, then scaled back to the requested size.

### Resource Identifiers

Negative icon numbers may be used to identify a group icon resource within an executable file. For example, the following sets the tray icon to the default icon used by ahk files:

```
Menu, Tray, Icon, %A_AhkPath%, -160
```

# AutoHotkey\_H New Features

[\[home\]](#)

| Commands / Functions                      |                                                                                                                                                                               |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Cast</a>                      | Converts a value from one data type to another data type.                                                                                                                     |
| <a href="#">Macros</a>                    | Create a macro instead of a function. Macro will use caller's scope for all variables except for parameters.                                                                  |
| <a href="#">#DllImport</a>                | Create an alias function for a dll function that allows to omit parameters and use default values.<br>It can also load pure machine code supplied as hex string, see example. |
| <a href="#">#NoEnv / GetEnv</a>           | #NoEnv is now used in all scripts. Use GetEnv to add environment variables to script.                                                                                         |
| <a href="#">#WarnContinuableException</a> | Do not warn on continuable exception, continue execution like main AutoHotkey does.                                                                                           |
| <a href="#">AhkThread</a>                 | Create a new thread using AutoHotkey.dll included in resources.                                                                                                               |
| <a href="#">ExeThread</a>                 | Create a new thread 'without' using AutoHotkey.dll, it uses same methods as AhkExported.                                                                                      |
| <a href="#">ThreadObj</a>                 | Create a new thread 'without' using AutoHotkey.dll, based on COM using ObjShare.                                                                                              |
| <a href="#">NewThread</a>                 | Create a new thread 'without' using AutoHotkey.dll. If you want to communicate with the thread it is best to use ExeThread or ThreadObj.                                      |
|                                           |                                                                                                                                                                               |

|                           |                                                                                                                                                                                                                                                                            |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alias                     | Create ByRef variable or share variables to other threads in a multi-thread environment.                                                                                                                                                                                   |
| BinToHex                  | Convert binary data to Hex string.                                                                                                                                                                                                                                         |
| BinRun                    | Run executable file from Memory.                                                                                                                                                                                                                                           |
| CreateScript              | Create script from main script for NewThread, AutoHotkey.dll, DynaRun or BinRun.                                                                                                                                                                                           |
| CriticalSection           | Creates a <a href="#">Critical Section Structure</a> and returns its pointer. Use EnterCriticalSection and LeaveCriticalSection to Enter and Leave the Critical Section.<br>Critical Section is required for multi-threading environment.                                  |
| DirGetParent              | Get parent directory of a file or directory.                                                                                                                                                                                                                               |
| ErrorMessage              | Get error message string of A_LastError.                                                                                                                                                                                                                                   |
| ExtractIconFromExecutable | Extract an icon from exe or dll.                                                                                                                                                                                                                                           |
| FileReplace               | Similar to FileAppend but replaces the file if it exists.                                                                                                                                                                                                                  |
| FindFunc (low level)      | Get a pointer to a Function.                                                                                                                                                                                                                                               |
| FindLabel (low level)     | Get a pointer to a label.                                                                                                                                                                                                                                                  |
| GetEnv                    | Get environment variables and make global variables in script.                                                                                                                                                                                                             |
| HexToBin                  | Convert Hex string to binary memory.                                                                                                                                                                                                                                       |
| HIBYTE                    | Get high byte from a value.                                                                                                                                                                                                                                                |
| HIWORD                    | Get high word from a value.                                                                                                                                                                                                                                                |
| Input                     | While Input is running the variable will be updated instantly and will be available to script. Use SetTimer or AutoHotkey.dll to get content of variable while Input is running. New option A is available to append input to variable, otherwise the variable will be set |

|              |                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
|              | empty before Input starts.                                                                                                             |
| IsBom        | Check if a file has byte order mark.                                                                                                   |
| LoadPicture  | Load picture into memory and return HBITMAP.                                                                                           |
| LOBYTE       | Get low byte from a value.                                                                                                             |
| LOWORD       | Get low word from a value.                                                                                                             |
| MAKELANGID   | Make LANGID.                                                                                                                           |
| MAKELCID     | Make LCID.                                                                                                                             |
| MAKELONG     | Make LONG.                                                                                                                             |
| MAKELPARAM   | Make LPARAM.                                                                                                                           |
| MAKELRESULT  | Make LRESULT.                                                                                                                          |
| MAKEWORD     | Make WORD.                                                                                                                             |
| MAKEWPARAM   | Make WPARAM.                                                                                                                           |
| MCodeH       | Create machine code function.                                                                                                          |
| OnMessage    | New parameter allows monitoring same message number for multiple windows and call different functions for each window.                 |
| Progress     | Progress function as in AutoHotkey v1.                                                                                                 |
| ResDelete    | Delete a resource in executable file.                                                                                                  |
| ResDllCreate | Create a resource only dll file.                                                                                                       |
| ResExist     | Check if a resource exists in executable file.                                                                                         |
| ResPut       | Add a resource to executable file from memory.                                                                                         |
| ResPutFile   | Add a file to resources of executable file.                                                                                            |
| Send         | Sleep functionality for Send command, for example <b>Send 123{100}456</b> would send 123 then sleep for 100 milliseconds and send 456. |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sizeof             | Returns the size in bytes for <a href="#">default data type</a> , <a href="#">structure</a> or <a href="#">structure definition</a> , for example MsgBox % sizeof("TCHAR").                                                                                                                                                                                                                                                                                                          |
| SplashImage        | SplashImage as in AutoHotkey v1.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| SplashTextOn / Off | SplashTextOn as in AutoHotkey v1.                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| StrPutVar          | Put an encoded string into a variable.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| ToChar             | Convert an integer to signed char.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ToInt              | Convert an integer to signed integer.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| ToShort            | Convert an integer to signed short.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| ToUChar            | Convert an integer to unsigned char.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ToUInt             | Convert an integer to signed integer.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| ToUShort           | Convert an integer to unsigned integer.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| VarSetCapacity     | When the buffer size of a variable is changed and FillByte is not used and old and new capacity is > 1 byte, the original buffer content is kept/copied to the new buffer.                                                                                                                                                                                                                                                                                                           |
| WinApi             | Use WinApi functions as if it was a build-in function.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Zip Functions      | Various zip functions to compress and uncompress files or data in memory.<br><a href="#">ZipCreateFile</a> , <a href="#">ZipAddFile</a> , <a href="#">ZipCloseFile</a> , <a href="#">UnZip</a> , <a href="#">ZipCreateBuffer</a> , <a href="#">ZipAddBuffer</a> , <a href="#">ZipCloseBuffer</a> , <a href="#">UnZipBuffer</a> , <a href="#">ZipRawMemory</a> , <a href="#">UnZipRawMemory</a> , <a href="#">ZipInfo</a> , <a href="#">ZipOptions</a> , <a href="#">ZipAddFolder</a> |
| Objects            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| ComObjDll          | Creates a COM Object from a COM dll.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| CriticalObject     | Create multi-thread save object.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| DynaCall           | Imports a Dll function and defines its parameters so we don't need to define on each                                                                                                                                                                                                                                                                                                                                                                                                 |

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
|                      | call like in DllCall.                                                                                                                         |
| ObjByRef             | Save variables by reference in object.                                                                                                        |
| ObjDump              | Dump an object to memory or file.                                                                                                             |
| ObjLoad              | Load a dumped object from memory or file.                                                                                                     |
| ObjShare             | Create multi-thread save COM IDispatch proxy object.                                                                                          |
| Struct               | Creates C++ like Structure object that is accessed and modified using object syntax.                                                          |
| <b>MemoryModule</b>  | <b>Allows loading a dll from Memory, e.g. Resource.</b>                                                                                       |
| MemoryLoadLibrary    | Similar to <a href="#">LoadLibrary</a> but loads a dll from memory. This allows real multi-threading by loading loading a dll multiple times. |
| MemoryGetProcAddress | Similar to <a href="#">GetProcAddress</a> for MemoryModule.                                                                                   |
| MemoryFreeLibrary    | Similar to <a href="#">FreeLibrary</a> for MemoryModule.                                                                                      |
| MemoryFindResource   | Similar to <a href="#">FindResource</a> .                                                                                                     |
| MemorySizeofResource | Similar to <a href="#">SizeOfResource</a> for MemoryModuel.                                                                                   |
| MemoryLoadResource   | Similar to <a href="#">LoadResource</a> .                                                                                                     |
| MemoryLoadString     | Similar to <a href="#">LoadString</a> .                                                                                                       |
| <b>Variables</b>     |                                                                                                                                               |
| NULL                 | New built-in, variable same as FALSE. Resolves to 0.                                                                                          |
| A_AhkDir             | returns the path of current exe.                                                                                                              |
| A_IsDll              | returns 1 (true) if called from AutoHotkey.dll and FALSE/NULL/0 otherwise.                                                                    |
| A_CoordModeToolTip   | returns current CoordMode for ToolTip, 0 =                                                                                                    |

|                       |                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | Client   1 = Window   2 = Screen.                                                                                                                                                                                                                                                                                                                   |
| A_CoordModePixel      | returns current CoordMode for Pixel, 0 = Client   1 = Window   2 = Screen.                                                                                                                                                                                                                                                                          |
| A_CoordModeMouse      | returns current CoordMode for Mouse, 0 = Client   1 = Window   2 = Screen.                                                                                                                                                                                                                                                                          |
| A_CoordModeCaret      | returns current CoordMode for Caret, 0 = Client   1 = Window   2 = Screen.                                                                                                                                                                                                                                                                          |
| A_CoordModeMenu       | returns current CoordMode for Menu, 0 = Client   1 = Window   2 = Screen.                                                                                                                                                                                                                                                                           |
| A_DllPath             | returns the path of current module (dll or exe).                                                                                                                                                                                                                                                                                                    |
| A_DllDir              | returns the path of current module (dll or exe).                                                                                                                                                                                                                                                                                                    |
| A_ModuleHandle        | Equivalent to GetModuleHandle(NULL) but returns the correct ModuleHandle for AutoHotkey.dll when using <a href="#">MemoryModule</a>                                                                                                                                                                                                                 |
| A_ScriptStruct        | returns pointer to internal g_script structure.                                                                                                                                                                                                                                                                                                     |
| A_GlobalStruct        | returns pointer to internal g structure.                                                                                                                                                                                                                                                                                                            |
| A_MainThreadID        | returns ID of main exe or dll thread.                                                                                                                                                                                                                                                                                                               |
| A_ThreadID            | returns ID of current thread.                                                                                                                                                                                                                                                                                                                       |
| A_ZipCompressionLevel | Get or set compression level for Zip functions. Use 0 for lowest and 9 for highest compression.                                                                                                                                                                                                                                                     |
| <b>Compiling</b>      |                                                                                                                                                                                                                                                                                                                                                     |
| Compiling AutoHotkey  | Original AutoHotkey is only capable to be compiled with AutoHotkeySC.bin. In AutoHotkey_H any AutoHotkey binary (AutoHotkey.dll, AutoHotkey.exe, AutoHotkeySC.bin) can be compiled. This allows keeping full functionality of AutoHotkey including executing other scripts. Compiled AutoHotkey.exe and AutoHotkey.dll can use /E switch to execute |

|                                                     |                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                     | different script than compiled one.                                                                                                                                                                                                                                                                                                           |
| <a href="#">Ahk2Exe</a>                             | Custom version of finsc <a href="#">Ahk2Exe</a> compiler with support for AutoHotkey_H features.                                                                                                                                                                                                                                              |
| Resource Library                                    | You can include library functions in resource of exe or dll (Resource type must be "LIB", e.g. LIB/WATCHDIRECTORY.AHK).<br>Additionally AutoHotkey.dll can load library functions from exe file automatically (Note both executables must be using same password if source is encrypted).<br>Note, this is not supported by AutoHotkeySC.bin. |
| <a href="#">Resource Compression and Encryption</a> | <a href="#">Ahk2Exe</a> supports compression and encryption for resource files, AutoHotkey will decompress and decrypt the files internally automatically. <a href="#">UnZipRawMemory</a> can be used to decompress and decrypt resources in Script.                                                                                          |
| AutoHotkey.dll Module                               |                                                                                                                                                                                                                                                                                                                                               |
| <a href="#">AutoHotkey.dll</a>                      | AutoHotkey Module with COM support that can be used for multi-threading and allows AutoHotkey to be embedded into another applications. It provides AutoHotkey functionality that might be more difficult to implement in another language.                                                                                                   |
| Exported Functions                                  | Available for AutoHotkey.exe and AutoHotkey.dll                                                                                                                                                                                                                                                                                               |
| <a href="#">ahkIsUnicode</a>                        | returns 1 (true) to identify that dll is Unicode.                                                                                                                                                                                                                                                                                             |
| <a href="#">ahkFunction</a>                         | Call a function via SendMessage method. Mainly used with AutoHotkey.dll to call a function in dll script or call a function in main script from dll.                                                                                                                                                                                          |
|                                                     |                                                                                                                                                                                                                                                                                                                                               |

|                    |                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| ahkPostFunction    | Call a function via PostMessage method (does not wait until function returns). Also used mainly with AutoHotkey.dll                    |
| ahkExecuteLine     | Executes script from given line pointer.                                                                                               |
| ahkLabel           | Goto (PostMessage) or Gosub (SendMessage) a Label. Also used mainly with AutoHotkey.dll                                                |
| ahkFindFunction    | Find a function and return its pointer.                                                                                                |
| ahkFindLabel       | Find a label and return its pointer.                                                                                                   |
| addFile            | Add and optionally execute additional script/code from file. Not available for scripts compiled with AutoHotkeySC.bin.                 |
| addScript          | Add and optionally execute additional script/code from text/memory/variable. Not available for scripts compiled with AutoHotkeySC.bin. |
| ahkExec            | Execute some script/code from text/memory/variable temporarily. Not available for scripts compiled with AutoHotkeySC.bin.              |
| ahkassign          | Assign a value to variable or pointer of variable.                                                                                     |
| ahkgetvar          | Retrieve a value from a variable.                                                                                                      |
| ahkPause           | Pause Script.                                                                                                                          |
| Exported Functions | Available only in AutoHotkey.dll                                                                                                       |
| ahkdll             | Load a new thread from a file, current thread will be terminated.                                                                      |
| ahktextdll         | Load a new thread from a string/memory/variable, current thread will be terminated.                                                    |
|                    |                                                                                                                                        |

|                           |                                                                                                                                                                                                                                                                                                     |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ahkReady</code>     | Returns 1 (true) if a thread is being executed currently, 0 (false) otherwise.                                                                                                                                                                                                                      |
| <code>ahkTerminate</code> | Terminate thread.                                                                                                                                                                                                                                                                                   |
| <code>ahkReload</code>    | Reload thread using same parameters used with <code>ahkdll</code> or <code>ahktextdll</code> .                                                                                                                                                                                                      |
| Other                     |                                                                                                                                                                                                                                                                                                     |
| <code>ahk_parent</code>   | Allow to identify the right window if multiple window with same criteria exist, <code>ahk_parent</code> must be followed by space and Window Id. <code>ahk_parent</code> can be used in any Win... and Control... functions. For example<br><b>WinActivate, ahk_class #32770 ahk_parent 0x3F4A5</b> |
| <code>&amp;</code>        | Retrieve pointer to any String or build in variable like <code>&amp;A_LoopField</code> or <code>&amp;"Text"</code> .                                                                                                                                                                                |

## Function Libraries

In addition to the user library in `%A_MyDocuments%\AutoHotkey\Lib`, standard library in the AutoHotkey directory and local library which resides in `%A_ScriptDir%\Lib` functions may be auto-included from a folder that `%A_AhkExeDir%\lib.lnk` points to. For more information, see [Libraries of Functions](#).

## Static Variables

Static variables are saved in separate array internally for better performance. Also [ListVars](#) shows static and local variables separately.

# AutoHotkey Beginner Tutorial

## Table of Contents

1. The basics
  1. Downloading AutoHotkey
  2. How to create a script
  3. You cannot merge commands
  4. Other basic info
2. Hotkeys & Hotstrings
  1. Keys and symbols
  2. Window specific
  3. Multiple hotkeys per file
  4. Examples
3. Sending key strokes
  1. Games
4. Running programs & website
5. Commands vs. Functions()
  1. Code blocks
6. Variables
  1. When to use percents
  2. Getting user input
  3. Other Examples?
7. Objects
  1. Creating Objects

2. Using Objects

8. Other helpful goodies

1. The mysterious []'s

2. Finding your AHK version

3. Trial and Error

4. Indentation

5. Asking for Help

6. Other links

# 1 - The Basics

Before we begin our journey, let me give some advice. Throughout this tutorial you will see a lot of text and a lot of code. For optimal learning power, it is advised that you read the text and **try** the code. Then study the code.

You can copy and paste most examples on this page.

If you get confused, try reading the section again.

## a. Downloading and installing AutoHotkey

Before learning to use AutoHotkey (AHK), you will need to download it. After downloading it, you may possibly need to install it. But that depends on the version you want. For this guide we will use the Installer since it is easiest to set up.

Currently there is no installer for AutoHotkey v2. The easiest way to get it work is to **download the binaries**, follow the installation steps below, and replace AutoHotkey.exe with the downloaded binaries inside the installation folder.

### Text instructions:

1. Go to the AutoHotkey Homepage. <https://autohotkey.com/>
2. Click Download. <https://autohotkey.com/download/ahk-install.exe>
3. During installation of AutoHotkey, you will be asked to choose from

UNICODE or ANSI. In short, you would probably want to choose UNICODE. It has support for non-English letters and numbers (characters). Keep going until you see an Install button.

4. Once done, great! Continue on to section b.

**Video instructions:**

Frankie's "Install and Hello World"

<http://www.autohotkey.com/forum/viewtopic.php?t=77674>

## **b. How to create a script**

Once you have AutoHotkey installed, you will probably want it to do stuff. AutoHotkey is not magic, we all wish it was, but it is not. So we will need to tell it what to do. This process is called "Scripting".

**Text instructions:**

- 1. Right-Click on your desktop.
  - 2. Find "New" in the menu.
  - 3. Click "AutoHotkey Script" inside the "New" menu.
  - 4. Give the script a new name. Note: It must end with a .ahk extension. Ex. MyScript.ahk
  - 5. Find the newly created file on your desktop and Right-Click it.
  - 6. Click "Edit Script".
  - 7. A window should have popped up, probably Notepad. If so, SUCCESS!
-

So now that you have created a script, we need to add stuff into the file. For a list of all built-in commands, function and variables, see [section 5](#).

Here is a very basic script containing a Hotkey which types text using the [Send](#) command when the hotkey is pressed.

```
^j::
 Send, My First Script
Return
```

We will get more in-depth later on. Until then, here's an explanation of the above code.

- The first line. `^j::` is the Hotkey. `^` means `CTRL`, `j` is the letter `j`.

Anything to the **left** of `::` are the keys you need to press.

- The second line. `Send, My First Script` is how you SEND keystrokes. `SEND` is the command, anything after the comma (,) will be typed.

- The third line. `Return`. Return will become your best friend. It literally STOPS code from going any further, to the lines below. This will prevent many issues when you start having a lot of stuff in your scripts.

- 
- 8. Save the File.
  - 9. Double-Click the file/icon in the desktop to run it. Open notepad or (anything you can type in) and press `Ctrl` and `J`.
  - 10. Hip Hip Hooray! Your first script is done. Go get some reward snacks then return to reading the rest of this tutorial.

Video instructions:

Frankie's "Install and Hello World"

<http://www.autohotkey.com/forum/viewtopic.php?t=77674>

### c. You cannot merge commands

When you are making your code, you might have the urge to put several commands on the same line or inside of each other, don't. In section 5 we'll talk about why it doesn't work as you might expect and what you can do instead.

### d. Other basic info

How to find the Help File on your computer:

Downloads for v2.0-a076 and later include an offline help file. Currently there is no installer for v2, so the help file is wherever you put it.

Look for **AutoHotkey.chm** or a file that says AutoHotkey and has a yellow question mark on it.

Online Links:

[Documentation](#)

[Command List](#)

[Functions](#)

[Variables](#)

## 2 - Hotkeys & Hotstrings

What is a Hotkey? A hotkey is a key that is hot to the touch. ... Just kidding. It is a key or key combination that the person at the keyboard presses to trigger some actions.

What is a Hotstring? Hotstrings are mainly used to expand abbreviations as you type them (auto-replace), they can also be used to launch any scripted action.

Here is a hotkey:

```
^j::
 Send, My First Script
Return
```

Here is a hotstring:

```
::ftw::Free the whales
```

The difference between the two examples is that the hotkey will be triggered when you press **CTRL & J** while the **hotstring** will convert your typed "ftw" into "Free the whales".

*"So, how exactly does a person such as myself create a hotkey?"* Good question.

A hotkey is created by using a single pair of ::'s. The key or key combo needs to go on the **left** of the **::**. And the content needs to go below, followed by a **Return**.

**Note:** There are exceptions, but those tend to cause confusion a lot of the

time. So it won't be covered in the tutorial, at least, not right now.

```
esc::
 MsgBox Escape!!!!
Return
```

A hotstring has a pair of ::'s on each side of the text you want to trigger the text replacement. While the text to replace your typed text goes on the **right** of the second pair of ::'s.

Hotstring, as mentioned above, can also launch scripted actions. That's fancy talk for *"do pretty much anything"*. Same with hotkeys.

```
::btw::
 MsgBox You typed "btw".
Return
```

A nice thing to know is that you can have many lines of code for each hotkey, hotstring, label, and a lot of other things we haven't talked about yet.

```
^j::
 MsgBox Wow!
 MsgBox this is
 Run, Notepad.exe
 winactivate, Untitled - Notepad
 WinWaitActive, Untitled - Notepad
 send, 7 lines{!}{enter}
 sendinput, inside the ctrl{+}j hotkey
Return
```

## a. Keys and their mysterious symbols

You might be wondering "How the crud am I supposed to know that ^ means CTRL?!". Well, good question. To help you learn what ^ and other symbols mean, gaze upon this chart:

| Symbol | Description                                                                                          |
|--------|------------------------------------------------------------------------------------------------------|
| #      | Win (Windows logo key)                                                                               |
| !      | Alt                                                                                                  |
| ^      | Control                                                                                              |
| +      | Shift                                                                                                |
| &      | An ampersand may be used between any two keys or mouse buttons to combine them into a custom hotkey. |

(For the full list of symbols, see the [Hotkey](#) page)

Additionally, here is a list of all/most hotkey names that can be used on the **left** side of a hotkeys :: symbol:

[KeyList.htm](#)

You can define a custom combination of two (and only two) keys (except joystick buttons) by using & between them. In the below example, you would hold down `Numpad0` then press the second key to trigger the hotkey:

```
Numpad0 & Numpad1::
 MsgBox You pressed Numpad1 while holding
 down Numpad0.
Return
```

```
Numpad0 & Numpad2::
```

```
Run Notepad
Return
```

But you are now wondering if hotstrings have any cool modifiers since hotkeys do. Yes, they do!

Hotstrings modifiers go between the first set of ::'s. Such as:

```
::*:ftw::Free the whales
```

For additional hotkey and hotstring modifiers, information and examples, visit:

[Hotkeys](#)

[Hotstrings](#)

## **b. Window specific hotkeys/hotstrings**

Sometime you might want a hotkey or hotstring to only work (or be disabled) in a certain window. To do this, you will need to use either of these fancy commands with a # in-front of them.

```
#IfWinActive
```

```
#IfWinExist
```

These special commands (technically called "directives") create context-sensitive hotkeys and hotstrings. Simply specify a window title. But in some cases you might want to specify an HWND, group, or class. Those are a bit advanced and are covered more in-depth here: [#IfWinActive](#).

```
#IfWinActive Untitled - Notepad
#space::
 MsgBox You pressed Win+Spacebar in Notepad.
Return
#IfWinActive
```

To turn off context sensitivity, specify any #IfWin command but leave all of its parameters blank. For example:

```
; Notepad
#IfWinActive untitled - Notepad
!q::
 MsgBox, You pressed Alt and Q in Notepad.
Return
#IfWinActive

; Any window that isn't Untitled - Notepad
!q::
 MsgBox, You pressed Alt and Q in any window.
Return
```

When #IfWin commands are turned off (or never used in a script), all hotkeys and hotstrings are enabled for all windows.

The #IfWin commands are positional: they affect all hotkeys and hotstrings physically beneath them in the script.

```
; Notepad
#IfWinActive ahk_class Notepad
#space::
```

```
 MsgBox, You pressed Win+Spacebar in Notepad.
Return
::msg::You typed msg in Notepad
#IfWinActive

; MSPaint
#IfWinActive untitled - Paint
#space::
 MsgBox, You pressed Win+Spacebar in MSPaint!
Return
::msg::You typed msg in MSPaint!
#IfWinActive
```

For more in-depth information and similar commands, check out:

[#IfWinActive](#)

### c. Multiple hotkeys/hotstrings per file

This, for some reason crosses some people's minds. So I'll set it clear:

AutoHotkey has the ability to have *as many* hotkeys and hotstrings in 1 file as you want. Whether it's 1, or 3253 (or more).

```
#i::
 run, http://www.google.com/
Return

^p::
 run, notepad.exe
Return

~j::
 send, ack
```

Return

```
:*:acheiv::achiev
::achievement::achievement
::acquaintance::acquaintance
*:adquir::acquir
::aquisition::acquisition
*:agrat::agrat
*:align::align
::ameria::America
```

The above code is perfectly acceptable. Multiple hotkeys, multiple hotstrings.  
All in one big happy script file.

## d. Examples

```
::btw::By the way ;
Replaces "btw" with "By the way" as soon as you
press an EndChar.
*:btw::By the way ;
Replaces "btw" with "By the way" without
needing an EndChar

^n:: ;
Ctrl & n Hotkey ;
run, notepad.exe ;
Run the program notepad.exe when you press Ctrl
& n ;
Return ;
This ends the hotkey. The code below this will
not get triggered.

^b:: ;
Ctrl & b Hotkey ;
```

```
send, {ctrl down}c{ctrl up}
```

```
;
```

Copies the selected text. ^c could be used as well, but this method is more secure.

```
SendInput, [b]{ctrl down}v{ctrl up}[/b]
```

```
;
```

Wraps the selected text in bbcode (forum) Bold tags.

```
Return
```

```
;
```

This ends the hotkey. The code below this point will not get triggered.

## 3 - Sending key strokes

So now you decided that you want to send (type) keys to a program. We can do that. Use the [Send](#) command. Send literally sends keystrokes, to simulate typing or pressing of keys.

Before we get into things, here are some common issues that people have:

Just like Hotkeys, Send has special keys too. [Lots and lots of them.](#)

Here are the 4 most common symbols:

---

**!**: Sends the ALT key. For example, *Send This is text!a* would send the keys "This is text" and then press ALT+a. **Note:** !A produces a different effect in some programs than !a. This is because !A presses ALT+SHIFT+A and !a presses ALT+a. If in doubt, use lowercase.

**+**: Sends the SHIFT key. For example, *Send +abC* would send the text "AbC", and *Send !+a* would press ALT+SHIFT+a.

**^**: Sends the CONTROL (Ctrl) key. For example, *Send ^!a* would press CTRL+ALT+a, and *Send ^{Home}* would send CONTROL+HOME. **Note:** ^A produces a different effect in some programs than ^a. This is because ^A presses CONTROL+SHIFT+A and ^a presses CONTROL+a. If in doubt, use lowercase.

**#**: Sends the WIN key (the key with the Windows logo) therefore *Send #e* would hold down the Windows key and then press the letter "e".

*The next couple of paragraphs are talking about the [table on send page.](#)*

Note:

This table **does not** apply to **hotkeys**. Meaning, you do not wrap **CTRL** **or** **ENTER** (or any other key) inside {}'s when making a hotkey.

An example showing what shouldn't be done to a hotkey:

```
; When making a hotkey...
; WRONG
{LCtrl}::
 send, AutoHotkey
Return

; CORRECT
LCtrl::
 send, AutoHotkey
Return
```

The gigantic table above shows pretty much every special key built-in to AHK. Such as: **{enter}** and **{space}**.

A common issue lots of people have is they assume that the curly brackets are put in the documentation pages just for fun. But in fact **they are needed**. It's how AHK knows that **{!}** means "exclamation point" and not "press the **Alt** key". So please remember to check the table on the [send](#) page and make sure you have your brackets in the right places.

```
; Notice the ! is in {}'s? That's because if it
wasn't, AHK would
; press the ALT key.
```

```
send, This text has been typed{!}
```

```
; Same as above, but with the ENTER key. AHK
would type out "enter" if ...
```

```
; ... it wasn't wrapped in {}'s.
```

```
send, Multiple enter lines have enter been
sent. ; WRONG
```

```
send, Multiple{enter}lines have{enter}been
sent. ; CORRECT
```

Another common issue is that people think that **everything** needs to be wrapped in brackets with the send command. That is FALSE. If it's not in the chart, it does not need brackets. You do **not** need to wrap common letters, numbers or even some symbols (such as . (period)) in {}'s.

Also, with the Send commands you are able to send more than 1 letter, number or symbol at a time. So no need for a bunch of Send commands with 1 letter each.

```
; Don't wrap words or individual letters that
are not in the table mentioned above.
```

```
send, {a} ; WRONG
```

```
send, {b} ; WRONG
```

```
send, {c} ; WRONG
```

```
send, {a}{b}{c} ; WRONG
```

```
send, {abc} ; WRONG
```

```
send, abc ; CORRECT
```

To hold down or release a key, enclose the key name in brackets and then use the word UP or DOWN.

```
; This is how you hold 1 key down and press
```

```

another key (or keys).
; If 1 method doesn't work in your program,
please try the other.
send, ^s ; Both of these
send CTRL+s
send, {ctrl down}s{ctrl up} ; Both of these
send CTRL+s
Send, {ctrl down}c{ctrl up}
Send, {b down}{b up}
Send, {TAB down}{TAB up}
Send, {Up down} ; Press down the up-arrow key.
Sleep, 1000 ; Keep it down for one second.
Send, {Up up} ; Release the up-arrow key.

```

But now you are wondering *"How can I make my really long send commands readable?"*. Easy. Use what is known as a Continuation Section. Simply specify an opening parenthesis on a new line, then your content, finally a closing parenthesis on its own line. For more information, read about [Continuation Sections](#).

```

send,
(
Line 1
Line 2
Apples are a fruit.
)

```

**Note:** There are several different forms of send. Each has their own special features. If one form of send does not work for your needs, try another type of send. Simply replace the commands name "send" with "sendPlay" or whatever you want.

Here are most ways to send text:

Send

SendRaw

SendInput

SendPlay

SendEvent

For more information on what each one does, [read this](#).

## a. Games

### **This is important!**

A lot of games, especially modern ones, have cheat prevention software. Things like GameGuard, Hackshield, PunkBuster and several others. If a game has a cheat prevention system and your hotkeys, hotstrings and send commands do not work, you are out of luck.

Not only is bypassing these systems in violation of the games policies and will get you banned, they are complex to work around. There are methods that can increase the chance of working in some games, but there is no magical "*make it work in my game now!!!*" button. so try **ALL** of these before giving up.

There are also known issues with DirectX. If you are having issues and you know the game uses DirectX, try the stuff below. You should also try running the game in Windowed Mode, if possible. That fixes some DirectX issues.

More DirectX issues may occur when using pixel or image commands. Colors might turn out black (0x000000) no matter the color you try to get. That is another tricky thing to fix. Try running in Windowed Mode if you can.

There is no single solution to make AutoHotkey work in all programs. If everything you try fails, it may not be possible to use AutoHotkey for your needs.

From the [FAQ](#) page:

Some games use DirectInput exclusively. As a side-effect, they might ignore all simulated keystrokes and mouse clicks. To work around this, try one of the following (or a combination):

- Use [SendPlay](#) via: 1) the `SendPlay` command; 2) using [SendMode Play](#); and/or 3) the `hotstring` option `SP`.
- Increase `SetKeyDelay`. For example:
- `SetKeyDelay, 0, 50`
- `SetKeyDelay, 150, 150, Play`
- Try `ControlSend`, which might work in cases where the other `Send` modes fail.

## 4 - Running programs & websites

To run a program such as *Mspaint.exe*, *Calc.exe*, *script.ahk* or even a folder, you can use the `Run` command. It can even be used to open URLs such as <https://autohotkey.com/> . If your computer is setup to run the type of program you want to run, it's very simple:

```
; Run a program. Note: most programs will
require a FULL file path.
Run, %A_ProgramFiles%\Some_Program\Program.exe

; Run a website
Run, https://autohotkey.com
```

There are some other advanced features as well, such as Command-Line parameters and CLSID.

If you want to learn more about that stuff, visit the [run page](#).

Here are a few more samples:

```
; Several programs do not need a full path,
such as Windows-standard programs.
Run, Notepad.exe
Run, MsPaint.exe

; Run the "My Documents" folder using the
built-in AHK variable
Run, %A_MyDocuments%

; Run some websites
Run, https://autohotkey.com
Run, http://www.google.com
```

For more in-depth information and examples, check out:

[commands/Run.htm](#).

## 5 - Commands vs. Functions()

AutoHotkey has two main types of things used by the scripter to create code: Commands and Functions()

A list of all commands/functions: [commands/index.htm](https://www.autohotkey.com/docs/commands/index.htm)

Note that all commands can be called as functions and vice versa, except for control flow statements such as Return. Which syntax you want to use is up to you, but in general the function syntax is preferred to have more flexibility.

### Commands

You can tell what a command is by looking at its syntax (the way it looks). Commands do not use parenthesis "(" around the parameters like functions do. So a command would look like this:

```
Command, parameter1, parameter2, parameter3
```

When using commands, you cannot squish other commands onto the same line as a previous command.

You cannot put commands inside the parameters of other commands.

```
Msgbox, Hello Run, Notepad.exe ; Wrong
```

```
Msgbox, Hello, Run, Notepad.exe ; Wrong
```

```
Msgbox, Hello ; Correct
Run, Notepad.exe
```

Commands also differ from function in that they use "traditional syntax". Meaning: when you use a `variable`, you NEED to use %'s around it. `%variable%`. Any text and numbers do not need to be in "quotation marks". `This is some text`. Additionally, you cannot do math in the parameters, unlike functions().

You can do math in parameters if you force an expression with a single `%`, but that will not be covered.

## Functions

As stated above, functions are different because they use parenthesis. A typical function looks like:

```
Function(parameter1, parameter2, parameter3)
```

Functions have a few main differences:

1. You can do math in them.

```
-- SubStr(37*12, 1, 2)
```

```
-- SubStr(A_Hour-12, 2)
```

2. Variables do not need to be wrapped in percent signs.

```
-- SubStr(A_Now, 7, 2)
```

3. Functions can go inside of functions.

```
-- SubStr(A_AHKPath, inStr(A_AHKPath, "AutoHotkey"))
```

4. Text needs to be wrapped in quotes.

```
-- SubStr("I'm scripting, awesome!", 16)
```

Functions usually return a value differently than a command does. Commands need an *OutputVar* parameter, functions do not. The most common way to assign a variable to the value of a function is like so:

```
MyVariable:=Function(Parameters)
```

```
MyVariable:=SubStr("I'm scripting, awesome!",
16)
```

This isn't the only way, but it's the most common. You are assigning `MyVariable` to the value of the function (in this case, `SubStr(...)`) that is to the right of the `:=`.

[More about Functions](#)

In short:

```
; These are commands
Msgbox, This is some text.
StrReplace, Output, %Input%, AutoHotKey,
AutoHotkey
SendInput, This is awesome{!}{!}{!}

; These are Functions
MsgBox("This is some text.")
Output := StrReplace(Input, "AutoHotKey",
"AutoHotkey")
SendInput("This is awesome{!}{!}{!}")
```

## a. Code blocks

`Code blocks` are little curly brackets (`{` and `}`) that are there to group a section of

code together so that AutoHotkey knows it's one big family and that it needs to stay together. They are most often used with *If* and *Loops*. Without them, only the first line in the block is called.

In the following code, both lines are run only if var equals 5.

```
if (var=5)
{
 MsgBox, var equals %var%!!
 Exitapp
}
```

In the following code, the msgbox is only shown if var equals 5. The code will always exit, even if var **isn't** 5.

```
if (var=5)
 MsgBox, var equals %var%!!
Exitapp
```

This is perfectly fine since the if only had 1 line of code associated with it. It's exactly the same as above, but I outdented the second line so we know it's separate from the if.

```
if (var=5)
 MsgBox, var equals %var%!!
MsgBox, We are now 'outside' the if. We did not
need {}'s since there was only 1 line below it.
```

## 6 - Variables

Variables are like little post-it notes that hold some information. They can be used to store text, numbers, data from functions and commands or even mathematical equations. Without them, programming & scripting would be much more tedious.

Variables can be assigned a few ways, We'll cover the most common forms. Please pay attention to the colon equal sign (:=).

1. `variable := "text"`

This is the simplest form for a variable. Simply type in your text and done. Any text needs to be in "quotes".

2. `variable := variable2`

Same as above, but you are assigning a variable to a different variables value.

3. `variable := 6+8/3*2-sqrt(9)`

Thanks to expressions, you can do math!

All of them can be combined in two ways:

Method #1: `var:="The value of 5+ " Variable " is: " 5+Variable`

Method #2: `var:="The value of 5+%Variable% is: %5+Variable%"`

Any equal sign (=) with a symbol in front of it is called an **Assignment Operator**, which are always an expression. So := += -= .= etc. always use expressions.

## a. When to use percents

One of the most common issues with AutoHotkey involving variables is when to use the percent signs (%). Hopefully this will clear some confusion.

When to use %'s:

1. When you are using Commands (see above) you use percent signs.  
-- Except when the parameter is OutputVar.
2. When you want to use the content of a variable inside a quoted string (see the 2nd method in the previous section).

When **not** to use %'s:

1. In parameters that are output variables, For example: StrLen, OutputVar, %InputVar%
2. On the left side of an assignment: Var := "123abc"
3. Everywhere in expressions (but outside of quoted strings). For example:

```
If (Var1 != Var2)
 Var1 := Var2 + 100
```

## b. Getting user input

Sometimes you want to have the user to choose the value of stuff. There are several ways of doing this, but the simplest way is [Inputbox](#). Here is a simple example on how to ask the user a couple of questions and doing some stuff with what was entered.

```
InputBox, OutputVar, What is your first name?,
Question 1
if (OutputVar="Bill")
 MsgBox("That's is an awesome name,
%OutputVar%.")

InputBox, OutputVar2, Do you like AutoHotkey?,
Question 2
if (OutputVar2="yes")
 MsgBox("Thank you for answering
%OutputVar2%, %OutputVar%! We will become great
friends.")
else
 MsgBox("%OutputVar%, That makes me sad.")
```

### c. other examples?

```
Result := MsgBox("Would you like to
continue?",, 4)
if Result = "No"
 Return ; If No, stop the code
from going further.
MsgBox You pressed YES. ; Otherwise, the user
picked yes.
```

```
; Some examples showing when to use percents
and when not
```

```

Var := "text" ; Assign a var some
text.
VarNmbr := 6 ; Assign a var a
number.
Var2 := Var ; Assign a var to
another var.
Var3 := "%Var%" ; Assign a var to
another var using percents.
Var4 := Var ; Append a var to the
end of another var.
Var5 += VarNmbr ; Add the value of a
var to another var.
Var5 -= VarNmbr ; Subtract the value
of a var from another var.
Var6 := SubStr(Var, 2, 2) ; Var inside a
function.
Var7 := Var "Text" ; Assigns a var to
another var with some extra text.
Var8 := "%Var% Text" ; Assigns a var to
another var with some extra text using
percents.
MsgBox, %Var% ; Var inside a
command.
MsgBox(Var) ; Same as above, but
inside a function.
StrSplit, Var, %Var%, x ; Var inside a
command that uses InputVar and OutputVar.
if (VarNmbr = 6) ; Check if a var is
equal to a number.
if VarNmbr = 6 ; Same as above.
if (Var != VarNmbr) ; Check if a var is
not equal to another var.
if Var1 < Var2 ; Check if a var is
lesser than another var.

```

## 7 - Objects

**Objects** are a way of organizing your data for more efficient usage. Sometimes objects are referred to as arrays, but it's important to note that all arrays are just objects. We call objects different things depending on what we are using them for, but all objects are the same.

An object is basically a collection of variables. The variable names are known as "Keys", and the contents of the variables are "Values".

When you hear people calling an object an *array* or *indexed array*, it usually means that all the keys are sequential numbers 1 and up.

When you hear people calling an object an *associative array*, it means that the keys are either strings (text) or non-sequential numbers. Sometimes it's a mix of both, and sequential numbers too!

There are no restrictions to what a key or value can be, and they can even be other arrays!

When the values are arrays too, this is referred to as a *nested array*, and these will be explained later.

There are a number of reasons you might want to use an object for something. Some examples:

1. You want to have a numbered list of things, such as a grocery list (this would be referred to as an indexed array)

2. You want to represent a grid, perhaps for a board game (this would be done with nested objects)
3. You have a list of things where each thing has a name, such as the characteristics of a fruit (this would be referred to as an associative array)

## a. Creating Objects

There are a few ways to create an object, and the most common ones are listed below

1. `MyObject := ["one", "two", "three", 17]`

Bracket syntax. This will start you off with what is sometimes called an "indexed array". An indexed array is an object representing a list of items, numbered 1 and up. In this example, the value "one" is stored in object key 1 (aka index 1), and the value 17 is stored in object key 4 (aka index 4).

2. `Banana := {"Shape": "Elongated", "Color": "Yellow", "Taste": "Delicious", "Price": 3}`

Brace syntax. This will let you start off by defining what is sometimes called an "associative array". An associative array is a collection of data where each item has a name. In this example, the value "yellow" is stored in the object key "color". Also, the value 3 is stored in the object key "Price".

3. `MyObject := Array("one", "two", "three", 17)`

The "array" creation function. This is equivalent to the bracket syntax,

but wrapped in a function.

```
4. Banana := Object("Shape", "Elongated", "Color",
"Yellow", "Taste", "Delicious", "Price", 3)
```

The object creation function. This is equivalent to the brace syntax, but wrapped in a function.

It's important to remember that every one of these definitions all create the same thing (objects), just with different keys.

## b. Using Objects

There are many ways to use objects, including retrieving values, setting values, adding more values, and more.

### To set values:

Setting values in an object is as simple as setting the value of a variable. All you have to do is put your bracket or dot notation (as seen in the retrieval section) on the left side of an expression assignment symbol `:=`.

For example:

```
Banana.Consistency := "Mushy"
```

```
Banana["Pickled"] := True ; This banana has been
pickled. Eww.
```

### To retrieve values:

1. `Value := Banana["Color"]`

Bracket notation. This allows you to use an expression as the key to get the value from your object. In this case, I used the simple expression `"Color"`, which is (unsurprisingly) the key `Color`. You will get a message box with the word "Yellow", because that is what we set the key `Color` to in the [previous section](#).

2. `Value := Banana.Color`

Dot notation. This only lets you use literal strings for the keys. You cannot use variables in your keys with dot notation.

### To add new keys and values:

1. Directly adding values

To directly add a key and value, just set a key that doesn't exist yet. For example:

```
MyObject.NewKey := "Shiny"
```

```
MyObject["NewerKey"] := 3.1415
```

2. Inserting values

Another way to add keys and values to an object is to use one of the following methods.

```
MyObject.InsertAt(Index, Value1, Value2,
Value3...)
```

*Index* is any integer key. This will shift ALL higher integer keys up by the

number of values which were inserted, even if there are gaps (for example, only keys 1 and 100 exist, and you insert a value at key 50, it will shift 100 up to 101).

```
MyObject.Push(Value1, Value2, Value3...)
```

This "appends" the values to the end of the array *MyObject*. In other words, it inserts the values at the highest integer key plus one.

### To remove keys and values:

#### 1. Blanking the value out.

The simplest way to remove a value is to just blank it out. You can do this by setting it to "", also known as an *empty string*. This doesn't remove the key, but it will make the value appear identical to an unset value.

It is possible to tell that the key still exists by using the `HasKey` method, and it will still come up in a `For` loop. (for loops will be explained later)

#### 2. Removing the key

There are a few different ways to remove both the key *and* the value.

They are:

```
1. RemovedValue := MyObject.Delete(AnyKey)
```

The previous value of `MyObject[AnyKey]` will be stored in *RemovedValue*.

```
2. NumberOfRemovedKeys :=
```

```
MyObject.Delete(FirstKey, LastKey)
```

Using the remove method in this way allows you to remove a range of numbered/integer or string keys between FirstKey and LastKey.

The value it gives will be the number of keys that were removed, which is useful if you have a gap between your keys (eg, you specify keys 1 through four, but key number 2 doesn't exist, this will set NumberOfRemovedKeys to 3 as only three keys were there to be removed)

3. 

```
MyObject.Pop()
```

This removes the highest integer key, and returns the value. There are no keys higher than it to be affected.

4. 

```
RemovedValue := MyObject.RemoveAt(Index)
```

```
NumberOfRemovedKeys := MyObject.RemoveAt(Index,
Length)
```

This removes all keys from *Index* to *Index + Length - 1* (inclusive). If *Length* is omitted it defaults to 1. After removing the keys it takes all higher numbered/integer keys and moves them down to fill the gap, so that if there was a value at *Index + Length* it will now be at *Index*. This is similar to how the InsertAt method with multiple specified values works.

## 8 - Other helpful goodies

We have reached the end of our journey, my good friend. I hope you have learned something. But before we go, here are some other things that I think you should know. Enjoy!

### a. The mysterious []'s

Throughout the documentation, you will see these two symbols ([ and ]) surrounding code in the yellow syntax box at the top of almost all pages. Anything inside of these brackets are **OPTIONAL**. Meaning the stuff inside can be left out if you don't need them. When writing your code, it is very important to **NOT** type the []'s in your code.

On the [ControlGetText](#) page you will see this (without the colors):

```
ControlGetText, OutputVar [, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

So you could simply do this if you wanted:

```
ControlGetText, OutputVar
```

Or add in some more details:

```
ControlGetText, OutputVar, Control, WinTitle
```

What if you wanted to use ExcludeTitle but not fill in WinText or WinTitle?  
Simple!

```
ControlGetText, OutputVar, Control,,, ExcludeTitle
```

Please note that you cannot IGNORE parameters, you can however leave them blank.

If you were to Ignore "WinTitle, WinText", it would look like this and cause issues:

```
ControlGetText, OutputVar, Control, ExcludeTitle
```

This is valid.

```
ControlGetText, OutputVar, Control,,, ExcludeTitle
```

## b. Finding your AHK version

Run this code to see your AHK version:

```
MsgBox, %A_AHKVersion%
```

Or look for "AutoHotkey Help File" or "AutoHotkey.chm" in the start menu or your installation directory.

## c. Trial and Error

Trial and Error is a very common and effective way of learning. Instead of asking for help on every little thing, sometimes spending some time alone (sometimes hours or days) and trying to get something to work will help you learn faster.

If you try something and it gives you an error, study that error. Then try to fix your code. Then try running it again. If you still get an error, modify your code some more. Keep trying and failing until your code fails no more. You will learn a lot this way by reading the documentation, reading errors and learning what works and what doesn't. Try, fail, try, fail, try, try, try, fail, fail, **succeed!**

This is how a lot of "pros" have learned. But don't be afraid to ask for help, we don't bite (hard). Learning takes time, the "pros" you encounter did not learn to be masters in just a few hours or days.

"If at first you don't succeed, try, try, try again." - Hickson, William E.

## d. Indentation

This stuff (indentation) is very important! Your code will run perfectly fine without it, but it will be a major headache for you and other to read your code. Small code (25 lines or less) will probably be fine to read without indentation, but it'll soon get sloppy. It's best you learn to indent ASAP.

Indentation has no set style, but it's best to keep everything consistent.

"**What is indentation?**" you ask? It's simply spacing to break up your code so you can see what belongs to what. People usually use 3 or 4 spaces or 1 tab per "level".

No indents:

```
if (car="old")
{
```

```
msgbox, the car is really old
if (wheels="flat")
{
msgbox, this car is not safe to drive.
Return
}
else
{
msgbox, Be careful! This old car will be
dangerous to drive.
}
}
else
{
msgbox, My`, what a shiny new vehicle you have
there.
}
```

Indented:

```
if (car="old")
{
 msgbox, the car is really old
 if (wheels="flat")
 {
 msgbox, this car is not safe to drive.
 Return
 }
 else
 {
 msgbox, Be careful! This old car will be
dangerous to drive.
 }
}
else
{
```

```
 msgbox, My`, what a shiny new vehicle you
 have there.
}
```

Wiki has various styles and examples. Choose what you like or learn to indent how you think it's easiest to read.

[http://en.wikipedia.org/wiki/Indent\\_style](http://en.wikipedia.org/wiki/Indent_style)

## e. Asking for Help

Before you ask, try doing some research yourself or try to code it yourself. If that did not yield results that satisfy you, read below.

- Don't be afraid to ask for help, even the smartest people ask others for help.
- Don't be afraid to show what you tried, even if you think it's silly.
- Post anything you have tried.
- Pretend *everyone but you* is a doorknob and knows nothing. Give as much information as you can to educate us doorknobs at what you are trying to do. Help us help you.
- Be patient.
- Be polite.
- Be open.
- Be kind.
- Enjoy

If you don't get an answer right away, wait at least 1 day (24 hours) before asking for more help. We love to help, but we also do this for free on our own time. We might be at work, sleeping, gaming, with family or just too busy to

help.

And while you wait for help, you can try learning and doing it yourself. It's a good feeling, making something yourself without help.

## **f. Other links**

[Frequently Asked Questions \(FAQ\)](#)

# Changes & New Features

The change log for AutoHotkey\_L can be found on [github](#).

# Changes & New Features

The change log for AutoHotkey\_H can be found on [github](#).

Main AutoHotkey change log can be also found on [github](#).

# AutoHotkey Script Showcase

[NiftyWindows -- by Enovatic-Solutions](#): This script gives you easy control of all basic window interactions such as dragging, resizing, maximizing, minimizing and closing. Its most powerful feature is activated by dragging with the right mouse button. Visualize each window divided into a virtual 9-cell grid with three columns and rows. The center cell is the largest one: you can grab and move a window around by clicking and holding it with the right mouse button. The other eight cells are used to resize a window in the same manner. NiftyWindows also offers snap-to-grid, "keep window aspect ratio", rolling up a window to its title bar, transparency control, and other useful shortcuts.

[Screen Magnifier -- by Holomind](#): This screen magnifier has the several advantages over the one included with the operating system, including: Customizable refresh interval and zoom level (including shrink/de-magnify); antialiasing to provide nicer output; and it is open-source (as a result, there are several variations to choose from, or you can tweak the script yourself).

[LiveWindows: Watch Dialog-boxes in Thumbnail -- by Holomind](#): This script allows you to monitor the progress of downloads, file-copying, and other dialogs by displaying a small replica of each dialog and its progress bar (dialogs are automatically detected, even if they're behind other windows). The preview window stays always-on-top but uses very little screen space (it can also be resized by dragging its edges). You can also monitor any window by dragging a selection rectangle around the area of interest (with control-shift-drag), then press Win+W to display that section in the preview window with real-time

updates.

[Mouse Gestures -- by deguix](#): This script watches how you move the mouse whenever the right mouse button is being held down. If it sees you "draw" a recognized shape or symbol, it will launch a program or perform another custom action of your choice (just like hotkeys). See the included README file for how to define gestures.

[Context Sensitive Help in Any Editor -- by Rajat](#): This script makes Ctrl+2 (or another hotkey of your choice) show the help file page for the selected AutoHotkey command or keyword. If nothing is selected, the command name will be extracted from the beginning of the current line.

[Easy Window Dragging \(requires XP/2k/NT\)](#): Normally, a window can only be dragged by clicking on its title bar. This script extends that so that any point inside a window can be dragged. To activate this mode, hold down CapsLock or the middle mouse button while clicking, then drag the window to a new position.

[Easy Window Dragging -- KDE style \(requires XP/2k/NT\) -- by Joany](#): This script makes it much easier to move or resize a window: 1) Hold down the ALT key and LEFT-click anywhere inside a window to drag it to a new location; 2) Hold down ALT and RIGHT-click-drag anywhere inside a window to easily resize it; 3) Press ALT twice, but before releasing it the second time, left-click to minimize the window under the mouse cursor, right-click to maximize it, or middle-click to close it.

[Easy Access to Favorite Folders -- by Savage](#): When you click the middle mouse

button while certain types of windows are active, this script displays a menu of your favorite folders. Upon selecting a favorite, the script will instantly switch to that folder within the active window. The following window types are supported: 1) Standard file-open or file-save dialogs; 2) Explorer windows; 3) Console (command prompt) windows. The menu can also be optionally shown for unsupported window types, in which case the chosen favorite will be opened as a new Explorer window.

**IntelliSense -- by Rajat (requires XP/2k/NT):** This script watches while you edit an AutoHotkey script. When it sees you type a command followed by a comma or space, it displays that command's parameter list to guide you. In addition, you can press Ctrl+F1 (or another hotkey of your choice) to display that command's page in the help file. To dismiss the parameter list, press Escape or Enter.

**Using a Joystick as a Mouse:** This script converts a joystick into a three-button mouse. It allows each button to drag just like a mouse button and it uses virtually no CPU time. Also, it will move the cursor faster depending on how far you push the joystick from center. You can personalize various settings at the top of the script.

**Joystick Test Script:** This script helps determine the button numbers and other attributes of your joystick. It might also reveal if your joystick is in need of calibration; that is, whether the range of motion of each of its axes is from 0 to 100 percent as it should be. If calibration is needed, use the operating system's control panel or the software that came with your joystick.

**On-Screen Keyboard:** This script creates a mock keyboard at the bottom of your

screen that shows the keys you are pressing in real time. I made it to help me to learn to touch-type (to get used to not looking at the keyboard). The size of the on-screen keyboard can be customized at the top of the script. Also, you can double-click the tray icon to show or hide the keyboard.

[Minimize Window to Tray Menu](#): This script assigns a hotkey of your choice to hide any window so that it becomes an entry at the bottom of the script's tray menu. Hidden windows can then be unhidden individually or all at once by selecting the corresponding item on the menu. If the script exits for any reason, all the windows that it hid will be unhidden automatically.

[Changing MsgBox's Button Names](#): This is a working example script that uses a timer to change the names of the buttons in a MsgBox dialog. Although the button names are changed, the MsgBox's return value still requires that the buttons be referred to by their original names.

[Numpad 000 Key](#): This example script makes the special 000 key that appears on certain keypads into an equals key. You can change the action by replacing the `Send, =` line with line(s) of your choice.

[Using Keyboard Numpad as a Mouse -- by deguix](#): This script makes mousing with your keyboard almost as easy as using a real mouse (maybe even easier for some tasks). It supports up to five mouse buttons and the turning of the mouse wheel. It also features customizable movement speed, acceleration, and "axis inversion".

[Seek](#): Navigating the Start Menu can be a hassle, especially if you have installed

many programs over time. 'Seek' lets you specify a case-insensitive key word/phrase that it will use to filter only the matching programs and directories from the Start Menu, so that you can easily open your target program from a handful of matched entries. This eliminates the drudgery of searching and traversing the Start Menu.

[ToolTip Mouse Menu \(requires XP/2k/NT\)](#) -- by Rajat: This script displays a popup menu in response to briefly holding down the middle mouse button. Select a menu item by left-clicking it. Cancel the menu by left-clicking outside of it. A recent improvement is that the contents of the menu can change depending on which type of window is active (Notepad and Word are used as examples here).

[Volume On-Screen-Display](#): This script assigns hotkeys of your choice to raise and lower the master and/or wave volume. Both volumes are displayed as different color bar graphs.

[Window Shading \(roll up a window to its title bar\)](#) -- by Rajat: This script reduces a window to its title bar and then back to its original size by pressing a single hotkey. Any number of windows can be reduced in this fashion (the script remembers each). If the script exits for any reason, all "rolled up" windows will be automatically restored to their original heights.

[WinLIRC Client](#): This script receives notifications from [WinLIRC](#) whenever you press a button on your remote control. It can be used to automate Winamp, Windows Media Player, etc. It's easy to configure. For example, if WinLIRC recognizes a button named "VolUp" on your remote control, create a label named

VolUp and beneath it use the command `SoundSet +5` to increase the soundcard's volume by 5%.

[1 Hour Software -- by skrommel](#): This is a large collection of useful scripts, professionally presented with short descriptions and screenshots.

[Toralf's Scripts](#): This collection includes useful scripts such as:

- 1) [AHK Window Info](#): Reveals information on windows, controls, etc.
- 2) [Electronic Program Guide](#): Browses the TV programs/schedules of your region (supports several countries).
- 3) [Auto-Syntax-Tidy](#): Changes indentation and case of commands in a script to give it a consistent format/style.

[Sean's Scripts](#): Includes useful scripts such as:

- 1) [Network Download/Upload Meter](#): Displays the network download/upload KB in a small, always-on-top progress bar.
- 2) [StdoutToVar](#): Redirects the output of a command or application into one of the script's variables.
- 3) [Capture a Screen Rectangle](#): A callable function that captures a portion of the screen and saves it as a file (BMP/JPG/PNG/GIF/TIF). It can also capture transparent windows and the mouse cursor.
- 4) [Color Zoomer/Picker](#): Magnifies the area near the cursor, allowing a single pixel to be selected and its color identified.

[SKAN's Tips N Tricks](#): Contains sample code and techniques for achieving useful effects and often-requested capabilities.

[Scripts and Functions Forum \(Current\)](#): This is a searchable collection of many ready-to-run scripts and functions. Built and maintained by AutoHotkey users, this archive grows and improves daily.

[Scripts and Functions Forum \(Archive\)](#): This is an archive of an older forum containing many more scripts, but some scripts might not run as-is on AutoHotkey v1.1.

[-- Home --](#)

# Script Compatibility

Although many scripts written for AutoHotkey 1.0 do not require changes to run on AutoHotkey 1.1, some may function incorrectly due to necessary differences between the two versions. As the most problematic differences only affect advanced functionality like DllCall, most users do not need to be concerned.

AutoHotkey 1.1 is also known as "AutoHotkey\_L", while AutoHotkey 1.0 was retrospectively labelled "AutoHotkey Basic". Some older versions of AutoHotkey\_L used 1.0.\* version numbers, so for clarity, this document refers to the two branches of AutoHotkey by name rather than version number.

**Note:** Some of the most common problems are caused by changes required to support Unicode text, and can be avoided by simply using the ANSI version of AutoHotkey\_L.

## Basic

High impact:

- Certain syntax errors are no longer tolerated
- FileRead may return corrupt binary data
- Variable and function names do not allow [, ] or ?
- DPI scaling is enabled by default for GUIs

Medium impact:

- Transform's *Unicode* sub-command is unavailable on Unicode versions
- AutoHotkey.ahk is launched instead of AutoHotkey.ini
- SetFormat, Integer, **H** is case-sensitive
- A\_LastError is modified by more commands
- MsgBox's handles commas more consistently
- GUI's Owner option overrides additional styles
- SoundSet and SoundGet work better on Vista and later
- ~Tilde affects how custom modifier keys work
- `x & y::` causes both `x::` and `x up::` to fire when x is released

## Low impact:

- If *var is type* ignores the system locale by default
- GroupActivate sets ErrorLevel and GroupAdd's *Label* works differently
- Run and RunWait interpret *Target* differently
- Control-Z is not interpreted as end-of-file
- Compatibility mode may cause confusion
- A\_IsCompiled is always read-only
- Leading and trailing ``t` sequences are no longer discarded

## Advanced

- Unicode vs ANSI
  - VarSetCapacity
  - DllCall
  - NumPut / NumGet
- Pointer Size

## Basic

### Syntax Errors

Certain syntax errors which were tolerated by AutoHotkey Basic are not tolerated by AutoHotkey\_L. Most such errors can be easily corrected once they are identified. The following errors are detected immediately upon launching a script in AutoHotkey\_L, and must be corrected for the script to run:

- A space, tab or comma is required between each command and its parameters. For example, `MsgBox< foo` and `If!foo` are not tolerated.
- `Hotkey, IfSomething`, where *Something* is invalid, is not tolerated.

Some other syntax errors are detected while the script is running. These cause an error message to be displayed prior to exiting the current thread:

- **Common:** Unrecognized or badly formatted `GuiCreate`, `Gui.Show` or `Gui.Add` options.
- `GroupAdd` with a blank group name. Previously this caused the thread to *silently* exit.
- Gui option `+LastFoundExist` must not be combined with another option, since that would cause it to act the same as `+LastFound`.

Some other syntax errors are currently not detected, but cause problems with AutoHotkey\_L:

- `Auto-concat` with `|` is more selective, so some invalid expressions like

`12(34)` no longer work.

## FileRead

`FileRead` translates text between code pages in certain common cases and therefore might output corrupt binary data. To avoid this, add the `*C` option or use `FileOpen` instead.

## Variable and Function Names

The characters `[`, `]` and `?` are reserved for use in expressions, so are no longer valid in variable names. Consequently, `?` (used in ternary operations) no longer requires a space on either side. See also [object syntax](#).

Errors may or may not be detected automatically:

- If a script used these characters in variable names in expressions, generally the script will run without displaying an error message, but will misbehave as the characters will be interpreted as operators rather than as part of a variable name.
- If these characters are used in a double-deref (such as `Array%n%` where `n` contains one of the above characters), an error message is displayed when the double-deref is evaluated, while the script is running.
- If these characters are used in other contexts, such as on the left hand side of an assignment, in the name of a command's input/output variable or between `%percent%` signs, an error message is displayed and the script is prevented from launching.

## DPI Scaling

DPI scaling is enabled by default for script GUIs to ensure they scale according to the [system DPI setting](#). If enabled and the system DPI setting is not 96 (100%), positions and sizes accepted by or returned from Gui methods/properties are not compatible with other commands. To disable DPI scaling, use `Gui.Opt("-DPIScale")`.

## Transform

Some *Transform* sub-commands are altered or unavailable in Unicode versions of AutoHotkey\_L:

- *Transform, Unicode* is unavailable. To assign Unicode text to the clipboard, use a regular assignment. See also: [StrPut/StrGet](#).
- *Transform, HTML* supports additional features.

## Default Script

When AutoHotkey\_L is launched without specifying a script, an .ahk file is loaded by default instead of an .ini file. The name of this file depends on the filename of the current executable. For more details, see [Passing Command Line Parameters to a Script](#).

## SetFormat, Integer[Fast], H

When an uppercase H is used, hexadecimal digits A-F will also be in uppercase.

AutoHotkey Basic always uses lowercase digits. See [SetFormat](#).

## A\_LastError

The following commands now set `A_LastError` to assist with debugging: `FileAppend`, `FileRead`, `FileReadLine`, `FileDelete`, `FileCopy`, `FileMove`, `FileGetAttrib/Time/Size/Version`, `FileSetAttrib/Time`, `FileCreateDir`, `RegRead`, `RegWrite`, `RegDelete`. Using any of these commands causes the previous value of `A_LastError` to be overwritten.

## MsgBox

`MsgBox`'s smart comma handling has been changed to improve flexibility and consistency with all other commands. In most cases, `MsgBox` will just work as intended. In some rare cases, scripts relying on the old quirky behaviour may observe a change in behaviour. For instance:

```
; This is now interpreted as an expression
(Options) followed by text (Title)
; instead of as a single expression (Text) with
multiple sub-expressions:
MsgBox % x, y
; Parentheses can be added to force the old
interpretation:
MsgBox % (x, y)

; This now shows an empty dialog instead of the
text "0, Title":
MsgBox 0, Title
; These behave as expected in both AutoHotkey_L
and AutoHotkey Basic:
```

```
MsgBox 0, Title, % "" ; Shows an empty dialog
MsgBox 0`, Title ; Shows the text "0,
Title"

; This now shows an empty dialog instead of the
text ", Title":
MsgBox,, Title
```

## GUI's Owner option

Applying the `+Owner` option to a Gui also removes the `WS_CHILD` style and sets the `WS_POPUP` style. This may break scripts which used `+Owner` to set the parent window of a Gui *after* setting the styles.

## Sound Commands on Windows Vista and later

`SoundSet`, `SoundGet`, `SoundSetWaveVolume` and `SoundGetWaveVolume` have improved support for Windows Vista and later. Typical changes in behaviour include:

- Scripts affecting the whole system (as is usually intended) instead of just the script itself.
- Devices being numbered differently - each output or input is considered a separate device.

## ~Tilde and Custom Combination Hotkeys

As of v1.1.14, the `tilde prefix` affects how a key works when used as a modifier key in a custom combination.

## Custom Combinations and Down/Up Hotkeys

Except when the tilde prefix is used, if both a key-down and a key-up hotkey are defined for a custom modifier key, they will both fire when the key is released.

For example, `x & y:::` causes both `x:::` and `x up:::` to fire when x is released, where previously `x:::` never fired.

## If *var is type*

`If var is type` ignores the system locale unless `StringCaseSense`, `Locale` has been used.

## Window Groups

`GroupActivate` sets `ErrorLevel` to 1 if no window was found to activate or 0 otherwise. Previously, `ErrorLevel` was left unchanged.

`GroupAdd`'s `Label` parameter applies to the window group as a whole instead of to one particular window specification within the group. A discussion of this change can be found [on the forums](#). However, using this parameter is **not recommended**; check `ErrorLevel` after calling `GroupActivate` instead.

## Run / RunWait

`AutoHotkey_L` includes some enhancements to the way the `Run` and `RunWait` commands interpret the `Target` parameter. This allows some things that didn't work previously, but in some very rare cases, may also affect scripts which were

already working in AutoHotkey Basic. The new behaviour is as follows:

- If *Target* begins with a quotation mark, everything up to the next quotation mark is considered the action, typically an executable file.
- Otherwise the first substring which ends at a space and is either an existing file or ends in .exe, .bat, .com, .cmd or .hta is considered the action. This allows file types such as .ahk, .vbs or .lnk to accept parameters while still allowing "known" executables such as wordpad.exe to be launched without an absolute path as in previous versions.

## Control-Z

`Loop Read` and `FileReadLine` no longer interpret the character Control-Z (0x1A) as an end-of-file marker. Any Control-Z, even one appearing at the very end of the file, is loaded as-is. `FileRead` already ignored this character, so is not affected by this issue.

## Compatibility Mode

If `Compatibility mode` is set to Windows 95, 98/ME or NT4 in the properties of the EXE file used to run the script, the script may not behave correctly. This is because compatibility mode causes a specific version of Windows to be reported to the application, but AutoHotkey\_L omits support for these versions. For example, setting compatibility mode to Windows 95 or 98/ME will cause `MsgBox %A_OSVersion%` to report `WIN_NT4`.

## A\_IsCompiled

`A_IsCompiled` is defined as an empty string if the script has not been compiled. Previously it was left undefined, which meant that assignments such as `A_IsCompiled := 1` were valid if the script hadn't been compiled. Now it is treated as a read-only built-in variable in all cases.

## Escaped Whitespace

Escaped whitespace characters such as `\t` and `\n` are no longer trimmed from the beginning and end of each arg. For example, `StringReplace s, s, \t` is now valid and will remove all tab characters from `s`.

## Unicode vs ANSI

A text value is often referred to as a *string*, as each text value is stored as a sequence or *string* of characters. The numeric character code and size (in bytes) of each character depends on which version of AutoHotkey you are using: *Unicode* or *ANSI*. These details are typically important for scripts which do any of the following:

- Pass strings to external functions via [DllCall](#).
- Pass strings via [PostMessage/SendMessage](#).
- Manipulate strings directly via [NumPut/NumGet](#).
- Use [VarSetCapacity](#) to ensure a variable can hold a specific number of characters.

Scripts designed with one particular format in mind will often encounter problems when run on the wrong version of AutoHotkey. For instance, some scripts written for AutoHotkey Basic will function correctly on the ANSI version of AutoHotkey\_L but fail on Unicode versions. If you aren't sure which version you are using, run the following script:

```
MsgBox % A_IsUnicode ? "Unicode" : "ANSI"
```

**ANSI:** Each character is **one byte** (8 bits). Character codes above 127 depend on your system's language settings.

**Unicode:** Each character is **two bytes** (16 bits). Character codes are as defined by the [UTF-16](#) format.

*Semantic note:* Technically, some Unicode characters are represented by *two* 16-bit code units, collectively known as a "surrogate pair." Similarly, some ANSI code pages (commonly known as Double Byte Character Sets) contain some double-byte characters. However, for practical reasons these are almost always treated as two individual units (referred to as "characters" for simplicity).

## VarSetCapacity

VarSetCapacity sets the capacity of a var *in bytes*. To set a variable's capacity based on the number of characters, the size of a character must be taken into account. For example:

```
VarSetCapacity(ansi_var, capacity_in_chars)
VarSetCapacity(unicode_var, capacity_in_chars *
2)
VarSetCapacity(native_var, capacity_in_chars *
(A_IsUnicode ? 2 : 1))
VarSetCapacity(native_var,
t_size(capacity_in_chars)) ; see below
```

There are two main uses for VarSetCapacity:

1. Expand a variable to hold an estimated number of characters, to enhance performance when building a string by means of gradual concatenation. For example, `VarSetCapacity(var, 1000)` allows for 1000 bytes, which is only 500 characters on Unicode versions of AutoHotkey\_L. This could affect performance, but the script should still function correctly.
2. Resize a variable to hold a binary structure. If the structure directly contains

text, the format of that text must be taken into account. This depends on the structure - sometimes ANSI text will be used even in a Unicode version of AutoHotkey\_L. If the variable is too small, the script may crash or otherwise behave unpredictably (depending on how the structure is used).

## DllCall

When the "Str" type is used, it means a string in the native format of the current build. Since some functions may require or return strings in a particular format, the following string types are available:

|             | Char Size | C / Win32 Types                   | Encoding                                                                    |
|-------------|-----------|-----------------------------------|-----------------------------------------------------------------------------|
| <b>WStr</b> | 16-bit    | wchar_t*, WCHAR*, LPWSTR, LPCWSTR | UTF-16                                                                      |
| <b>AStr</b> | 8-bit     | char*, CHAR*, LPSTR, LPCSTR       | ANSI (the system default ANSI code page)                                    |
| <b>Str</b>  | --        | TCHAR*, LPTSTR, LPCTSTR           | Equivalent to <b>WStr</b> in Unicode builds and <b>AStr</b> in ANSI builds. |

If "Str" or the equivalent type for the current build is used as a parameter, the address of the string or var is passed to the function, otherwise a temporary copy of the string is created in the desired format and passed instead. As a general rule, "AStr" and "WStr" should not be used if the function writes a value into that parameter.

**Note:** "AStr" and "WStr" are equally valid for parameters and the function's return value.

In general, if a script calls a function via DllCall which accepts a string as a parameter, one of the following approaches must be taken:

1. If both Unicode (W) and ANSI (A) versions of the function are available, call the appropriate one for the current build. In the following example, "DeleteFile" is internally known as "DeleteFileA" or "DeleteFileW". Since "DeleteFile" itself doesn't really exist, DllCall automatically tries "A" or "W" as appropriate for the current build:

```
DllCall("DeleteFile", "Ptr", &filename)
DllCall("DeleteFile", "Str", filename)
```

In this example, `&filename` passes the address of the string exactly as-is, so the function must expect a string in the same format as the "Str" type. Note that "UInt" must be used in place of "Ptr" in AutoHotkey Basic, but the resulting code may not be 64-bit compatible.

**Note:** If the function cannot be found exactly as specified, AutoHotkey\_L appends the "A" or "W" suffix regardless of which DLL is specified.

However, AutoHotkey Basic appends the "A" suffix only for functions in User32.dll, Kernel32.dll, ComCtl32.dll, or Gdi32.dll.

2. If the function only accepts a specific type of string as input, the script may use the appropriate string type:

```
DllCall("DeleteFileA", "AStr", filename)
DllCall("DeleteFileW", "WStr", filename)
```

3. If the function must modify a string (in a non-native format), the script must allocate a buffer as described [above](#) and pass its address to the function. If the parameter accepts input, the script must also convert the input string to the appropriate format; this can be done using [StrPut](#).

## NumPut / NumGet

When NumPut or NumGet are used with strings, the offset and type must be correct for the given type of string. The following may be used as a guide:

```
; 8-bit/ANSI strings: size_of_char=1
type_of_char="Char"
; 16-bit/UTF-16 strings: size_of_char=2
type_of_char="UShort"
nth_char := NumGet(var, (n-1)*size_of_char,
type_of_char)
NumPut(nth_char, var, (n-1)*size_of_char,
type_of_char)
```

If `var` contains a string in the native format, the appropriate values may be determined based on the value of `A_IsUnicode`:

```
nth_char := NumGet(var, t_size(n-1), t_char())
NumPut(nth_char, var, t_size(n-1), t_char())

; Define functions for convenience and clarity:
t_char() {
 return A_IsUnicode ? "UShort" : "Char"
}
t_size(char_count:=1) {
 return A_IsUnicode ? char_count*2 :
```



## Pointer Size

Pointers are 4 bytes in 32-bit builds (including AutoHotkey Basic) and 8 bytes in 64-bit builds. Scripts using structures or DllCalls may need to account for this to run correctly on both platforms. Specific areas which are affected include:

- Offset calculation for fields in structures which contain one or more pointers.
- Size calculation for structures containing one or more pointers.
- Type names used with `DllCall`, `NumPut` or `NumGet`.

For size and offset calculations, use `A_PtrSize`. For `DllCall`, `NumPut` and `NumGet`, use the `Ptr` type where appropriate.

Remember that the offset of a field is usually the total size of all fields preceding it. Also note that handles (including types like `HWND` and `HBITMAP`) are essentially pointer-types.

```
/*
typedef struct _PROCESS_INFORMATION {
HANDLE hProcess; // Ptr
HANDLE hThread;
DWORD dwProcessId; // UInt (4 bytes)
DWORD dwThreadId;
} PROCESS_INFORMATION,
*LPPROCESS_INFORMATION;
*/
VarSetCapacity(pi, A_PtrSize*2 + 8) ; Ptr + Ptr
+ UInt + UInt
DllCall("CreateProcess", <omitted for brevity>
```

```
"Ptr", &pi, <omitted>)
hProcess := NumGet(pi, 0) ; Defaults
to "Ptr".
hThread := NumGet(pi, A_PtrSize) ;
dwProcessId := NumGet(pi, A_PtrSize*2,
"UInt")
dwProcessId := NumGet(pi, A_PtrSize*2 + 4,
"UInt")
```

# Alphabetical Command and Function Index

Click on a command or function name for details. Entries in **large font** are the most commonly used.

| Entry                   | Description                                                                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>{ ... } (Block)</b>  | A pair of braces denotes a block. Blocks are typically used with <b>functions</b> , <b>Else</b> , <b>Loop</b> , <b>While-loop</b> , and <b>IF-commands</b> . |
| <b>{ ... } / Object</b> | Creates a scriptable associative array.                                                                                                                      |
| <b>[ ... ] / Array</b>  | Creates a scriptable associative array with integer keys.                                                                                                    |
| <b>Abs</b>              | Returns the absolute value of <i>Number</i> .                                                                                                                |
| <b>ASin</b>             | Returns the arcsine (the number whose sine is <i>Number</i> ) in radians.                                                                                    |
| <b>ACos</b>             | Returns the arccosine (the number whose cosine is <i>Number</i> ) in radians.                                                                                |
| <b>ATan</b>             | Returns the arctangent (the number whose tangent is <i>Number</i> ) in radians.                                                                              |
| <b>Between</b>          | Checks whether a variable's contents are numerically or alphabetically between two values (inclusive).                                                       |
| <b>BinRun</b>           | Run a executable file (.exe) from memory.                                                                                                                    |
| <b>BinToHex</b>         | Convert binary memory to hex string.                                                                                                                         |
| <b>BlockInput</b>       | Disables or enables the user's ability to interact with the computer via keyboard and mouse.                                                                 |

|               |                                                                                                                                    |
|---------------|------------------------------------------------------------------------------------------------------------------------------------|
| Break         | Exits (terminates) a <code>loop</code> . Valid inside any kind of <code>loop</code> .                                              |
| Catch         | Specifies the code to execute if an exception is raised during execution of a <code>try</code> statement.                          |
| Ceil          | Returns <i>Number</i> rounded up to the nearest integer (without any <code>.00</code> suffix).                                     |
| Chr           | Returns the string (usually a single character) corresponding to the character code indicated by the specified number.             |
| Click         | Clicks a mouse button at the specified coordinates. It can also hold down a mouse button, turn the mouse wheel, or move the mouse. |
| ClipWait      | Waits until the <code>clipboard</code> contains data.                                                                              |
| ComObjActive  | Retrieves a registered COM object.                                                                                                 |
| ComObjArray   | Creates a <code>SAFEARRAY</code> for use with COM.                                                                                 |
| ComObjConnect | Connects a COM object's event sources to functions with a given prefix.                                                            |
| ComObjCreate  | Creates a COM object.                                                                                                              |
| ComObjDll     | Creates a COM object from a <code>dll</code> .                                                                                     |
| ComObject     | Wraps a value, <code>SafeArray</code> or COM object for use by the script or for passing to a COM method.                          |
| ComObjError   | Enables or disables notification of COM errors.                                                                                    |
| ComObjFlags   | Retrieves or changes flags which control a COM wrapper object's behaviour.                                                         |
| ComObjGet     | Returns a reference to an object provided by a COM component.                                                                      |
|               |                                                                                                                                    |

|                     |                                                                                                                                    |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------|
| ComObjQuery         | Queries a COM object for an interface or service.                                                                                  |
| ComObjType          | Retrieves type information from a COM object.                                                                                      |
| ComObjValue         | Retrieves the value or pointer stored in a COM wrapper object.                                                                     |
| Continue            | Skips the rest of the current <a href="#">loop</a> iteration and begins a new one. Valid inside any kind of <a href="#">loop</a> . |
| ControlAddItem      | Adds a new entry at the bottom of a ListBox or ComboBox.                                                                           |
| ControlChoose       | Sets the selection in a ListBox or ComboBox to be the Nth entry.                                                                   |
| ControlChooseString | Sets the selection in a ListBox or ComboBox to be the first entry whose leading part matches a string.                             |
| ControlClick        | Sends a mouse button or mouse wheel event to a control.                                                                            |
| ControlDeleteItem   | Deletes the Nth entry from a ListBox or ComboBox.                                                                                  |
| ControlEditPaste    | Pastes a string at the caret/insert position in an Edit control.                                                                   |
| ControlFindItem     | Returns the entry number of a ListBox or ComboBox given the entry's text.                                                          |
| ControlFocus        | Sets input focus to a given control on a window.                                                                                   |
| ControlGetChecked   | Returns 1 (true) if the checkbox or radio button is checked or 0 (false) if not.                                                   |
| ControlGetChoice    | Returns the name of the currently selected entry in a ListBox or ComboBox.                                                         |
|                     | Returns the column number in an Edit control                                                                                       |

|                                                  |                                                                                    |
|--------------------------------------------------|------------------------------------------------------------------------------------|
| <code>ControlGetCurrentCol</code>                | where the caret (text insertion point) resides.                                    |
| <code>ControlGetCurrentLine</code>               | Returns the line number in an Edit control where the caret (insert point) resides. |
| <code>ControlGetEnabled</code>                   | Returns 1 (true) if the control is enabled, or 0 (false) if disabled.              |
| <code>ControlGetFocus</code>                     | Retrieves which control of the target window has input focus, if any.              |
| <code>ControlGetHwnd</code>                      | Retrieves the window handle (HWND) of the specified control.                       |
| <code>ControlGetLine</code>                      | Returns the text of line <i>N</i> in an Edit control.                              |
| <code>ControlGetLineCount</code>                 | Returns the number of lines in an Edit control.                                    |
| <code>ControlGetList</code>                      | Retrieves a list of items from a ListView, ListBox, ComboBox, or DropDownList.     |
| <code>ControlGetPos</code>                       | Retrieves the position and size of a control.                                      |
| <code>ControlGetSelected</code>                  | Returns the selected text in an Edit control.                                      |
| <code>ControlGetStyle / ControlGetExStyle</code> | Retrieves an integer representing the style or extended style of the control.      |
| <code>ControlGetTab</code>                       | Returns the position number of the selected tab in a SysTabControl32.              |
| <code>ControlGetText</code>                      | Retrieves text from a control.                                                     |
| <code>ControlGetVisible</code>                   | Returns 1 (true) if the control is visible, or 0 (false) if hidden.                |
| <code>ControlHide</code>                         | Hides a control.                                                                   |
| <code>ControlHideDropDown</code>                 | Hides the drop-down window of a ComboBox if it is visible.                         |
| <code>ControlMove</code>                         | Moves or resizes a control.                                                        |
| <code>ControlSend / ControlSendRaw</code>        | Sends simulated keystrokes to a window or control.                                 |
|                                                  |                                                                                    |

|                                     |                                                                                                                                         |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| ControlSetChecked                   | Turns on (checks) or turns off (unchecks) a radio button or checkbox.                                                                   |
| ControlSetEnabled                   | Enables or disables a control.                                                                                                          |
| ControlSetStyle / ControlSetExStyle | Changes the style or extended style of a control, respectively.                                                                         |
| ControlSetTab                       | Selects the Nth tab in a SysTabControl32.                                                                                               |
| ControlSetText                      | Changes the text of a control.                                                                                                          |
| ControlShow                         | Shows a control if it was previously hidden.                                                                                            |
| ControlShowDropDown                 | Drops a ComboBox so that its choices become visible.                                                                                    |
| CoordMode                           | Sets coordinate mode for various commands to be relative to either the active window or the screen.                                     |
| Cos                                 | Returns the trigonometric cosine of <i>Number</i> .                                                                                     |
| CreateScript                        | Creates a script from main script that can be passed to AutoHotkey.dll, BinRun or DynaRun.                                              |
| Critical                            | Prevents the current thread from being interrupted by other threads.                                                                    |
| CriticalObject                      | Built-in function to enwrap an object for multi-thread use. Such objects can be used from multiple threads without causing a crash./td> |
| CriticalSection                     | Included function to create and initialize a Critical Section Object.                                                                   |
| CryptAES                            | Encrypt and Decrypt data.                                                                                                               |
| DateAdd                             | Adds or subtracts time from a date-time value.                                                                                          |
| DateDiff                            | Compares two date-time values and returns the difference.                                                                               |
|                                     | Determines whether invisible text in a                                                                                                  |

|                     |                                                                                                                                                                                       |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DetectHiddenText    | window is "seen" for the purpose of finding the window. This affects commands such as WinExist and WinActivate.                                                                       |
| DetectHiddenWindows | Determines whether invisible windows are "seen" by the script.                                                                                                                        |
| DirCopy             | Copies a folder along with all its sub-folders and files (similar to xcopy).                                                                                                          |
| DirCreate           | Creates a folder.                                                                                                                                                                     |
| DirDelete           | Deletes a folder.                                                                                                                                                                     |
| DirGetParent        | Retrieve parent directory for a file or folder.                                                                                                                                       |
| DirMove             | Moves a folder along with all its sub-folders and files. It can also rename a folder.                                                                                                 |
| DirSelect           | Displays a standard dialog that allows the user to select a folder.                                                                                                                   |
| DllCall             | Calls a function inside a DLL, such as a standard Windows API function.                                                                                                               |
| Download            | Downloads a file from the Internet.                                                                                                                                                   |
| Drive               | Ejects/retracts the tray in a CD or DVD drive, or sets a drive's volume label.                                                                                                        |
| DynaCall            | Build in function, similar to DllCall but works with DllCall structures and uses Object syntax. It is often faster than DllCall, easier to use and it saves a lot of typing and code. |
| DynaRun             | Run code dynamically in new AutoHotkey process.                                                                                                                                       |
| DriveGet            | Retrieves various types of information about the computer's drive(s).                                                                                                                 |
| Edit                | Opens the current script for editing in the associated editor.                                                                                                                        |
|                     | Specifies the command(s) to perform if an IF-                                                                                                                                         |

|                    |                                                                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Else               | statement evaluates to FALSE. When more than one command is present, enclose them in a <code>block</code> (braces).                        |
| EnvGet             | Retrieves an environment variable.                                                                                                         |
| EnvSet             | Writes a value to a <code>variable</code> contained in the environment.                                                                    |
| EnvUpdate          | Notifies the OS and all running applications that <code>environment variable(s)</code> have changed.                                       |
| Exit               | Retrieve the error message from <code>system message table</code> resources.                                                               |
| Exception          | Creates an <code>Exception</code> object.                                                                                                  |
| ExeThread          | Included function to start new exe thread (without using <code>AutoHotkey.dll</code> ).                                                    |
| Exit               | Exits the <code>current thread</code> . If the script is not <code>persistent</code> and this is the last thread, the entire script exits. |
| ExitApp            | Terminates the script unconditionally.                                                                                                     |
| Exp                | Returns $e$ (which is approximately 2.71828182845905) raised to the $N$ th power.                                                          |
| FileAppend         | Writes text to the end of a file (first creating the file, if necessary).                                                                  |
| FileCopy           | Copies one or more files.                                                                                                                  |
| FileCreateShortcut | Creates a shortcut ( <code>.lnk</code> ) file.                                                                                             |
| FileDelete         | Deletes one or more files.                                                                                                                 |
| FileEncoding       | Sets the default encoding for <code>FileRead</code> , <code>Loop Read</code> , <code>FileAppend</code> , and <code>FileOpen</code> .       |
| FileExist          | Checks for the existence of a file or folder and returns its attributes.                                                                   |

|                  |                                                                                        |
|------------------|----------------------------------------------------------------------------------------|
| FileInstall      | Includes the specified file inside the <b>compiled version</b> of the script.          |
| FileGetAttrib    | Reports whether a file or folder is read-only, hidden, etc.                            |
| FileGetShortcut  | Retrieves information about a shortcut (.lnk) file, such as its target file.           |
| FileGetSize      | Retrieves the size of a file.                                                          |
| FileGetTime      | Retrieves the datetime stamp of a file or folder.                                      |
| FileGetVersion   | Retrieves the version of a file.                                                       |
| FileInstall      | Includes the specified file inside the <b>compiled version</b> of the script.          |
| FileMove         | Moves or renames one or more files.                                                    |
| FileOpen         | Provides object-oriented file I/O.                                                     |
| FileRead         | Reads a file's contents into a <b>variable</b> .                                       |
| FileRecycle      | Sends a file or directory to the recycle bin, if possible.                             |
| FileRecycleEmpty | Empties the recycle bin.                                                               |
| FileReplace      | Deletes the file and writes text to the file after creating it.                        |
| FileSelect       | Displays a standard dialog that allows the user to open or save file(s).               |
| FileSetAttrib    | Changes the attributes of one or more files or folders. Wildcards are supported.       |
| FileSetTime      | Changes the datetime stamp of one or more files or folders. Wildcards are supported.   |
| Finally          | Ensures that a block of code is always executed after a <b>Try</b> statement finishes. |

|               |                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------|
| FindFunc      | Finds a function in currently executed script and returns a pointer to it.                        |
| FindLabel     | Finds a label in currently executed script and returns a pointer to it.                           |
| Floor         | Returns <i>Number</i> rounded down to the nearest integer (without any .00 suffix).               |
| For           | Repeats a series of commands once for each key-value pair in an object.                           |
| Format        | Formats a variable number of input values according to a format string.                           |
| FormatTime    | Transforms a <code>YYYYMMDDHH24MISS</code> timestamp into the specified date/time format.         |
| Func          | Retrieves a <a href="#">reference</a> to a function.                                              |
| GetKeyName    | Retrieves the name or text of a key.                                                              |
| GetKeyVK      | Retrieves the virtual key code of a key.                                                          |
| GetKeySC      | Retrieves the scan code of a key.                                                                 |
| GetKeyState   | Checks if a keyboard key or mouse/joystick button is down or up. Also retrieves joystick status.  |
| GetVar        | Retrieves a pointer to a variable.                                                                |
| GetKeyVK      | Retrieves the name/text, virtual key code of a key.                                               |
| Gosub         | Jumps to the specified label and continues execution until <a href="#">Return</a> is encountered. |
| Goto          | Jumps to the specified label and continues execution.                                             |
| GroupActivate | Activates the next window in a window group that was defined with <a href="#">GroupAdd</a> .      |
| GroupAdd      | Adds a window specification to a window group, creating the group if necessary.                   |

|                 |                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |                                                                                                                                                                                                                    |
| GroupClose      | Closes the active window if it was just activated by <a href="#">GroupActivate</a> or <a href="#">GroupDeactivate</a> . It then activates the next window in the series. It can also close all windows in a group. |
| GroupDeactivate | Similar to <a href="#">GroupActivate</a> except activates the next window <b>not</b> in the group.                                                                                                                 |
| GuiCreate       | Creates a new <a href="#">Gui</a> object, which is essential for creating and managing a GUI.                                                                                                                      |
| GuiCtrlFromHwnd | Retrieves the <a href="#">GuiControl</a> object of a GUI control associated with the specified HWND.                                                                                                               |
| GuiFromHwnd     | Retrieves the <a href="#">Gui</a> object of a GUI window associated with the specified HWND.                                                                                                                       |
| HexToBin        | Convert hex string to binary memory.                                                                                                                                                                               |
| HIBYTE          | Retrieves the high-order byte from the given value.                                                                                                                                                                |
| HIWORD          | Retrieves the high-order word from the given value.                                                                                                                                                                |
| Hotkey          | Creates, modifies, enables, or disables a hotkey while the script is running.                                                                                                                                      |
| if (expression) | Specifies the command(s) to perform if an <a href="#">expression</a> evaluates to TRUE.                                                                                                                            |
| ImageSearch     | Searches a region of the screen for an image.                                                                                                                                                                      |
| IniDelete       | Deletes a value from a standard format .ini file.                                                                                                                                                                  |
| IniRead         | Reads a value from a standard format .ini file.                                                                                                                                                                    |
| IniWrite        | Writes a value to a standard format .ini file.                                                                                                                                                                     |
| Input           | Waits for the user to type a string.                                                                                                                                                                               |
|                 |                                                                                                                                                                                                                    |

|             |                                                                                                                                                                                             |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| InputEnd    | Terminates any Input in progress in another thread.                                                                                                                                         |
| InputBox    | Displays an input box to ask the user to enter a string.                                                                                                                                    |
| InStr       | Searches for a given <i>occurrence</i> of a string, from the left or the right.                                                                                                             |
| IsBOM       | Returns true if given address contains a byte order mark.                                                                                                                                   |
| IsByRef     | Determines whether a ByRef parameter was supplied with a variable.                                                                                                                          |
| IsFileInUse | Returns true / 1 if FilePattern is in use and cannot be read / deleted / written.                                                                                                           |
| IsFunc      | Returns a non-zero number if the specified function exists in the script.                                                                                                                   |
| IsLabel     | Returns a non-zero number if the specified label exists in the script.                                                                                                                      |
| IsObject    | Returns a non-zero number if the specified value is an object.                                                                                                                              |
| KeyHistory  | Displays script info and a history of the most recent keystrokes and mouse clicks.                                                                                                          |
| KeyWait     | Waits for a key or mouse/joystick button to be released or pressed down.                                                                                                                    |
| ListHotkeys | Displays the hotkeys in use by the current script, whether their subroutines are currently running, and whether or not they use the <a href="#">keyboard</a> or <a href="#">mouse</a> hook. |
| ListLines   | Displays the script lines most recently executed.                                                                                                                                           |
| ListVars    | Displays the script's <a href="#">variables</a> : their names and current contents.                                                                                                         |
|             |                                                                                                                                                                                             |

|                           |                                                                                                                     |
|---------------------------|---------------------------------------------------------------------------------------------------------------------|
| LoadPicture               | Loads a JPG/GIF/BMP/ICO/etc. and returns an HBITMAP or HICON.                                                       |
| LOBYTE                    | Retrieves the low-order byte from the given value.                                                                  |
| Log                       | Returns the logarithm (base 10) of <i>Number</i> .                                                                  |
| LOWORD                    | Retrieves the low-order word from the given value.                                                                  |
| Ln                        | Returns the natural logarithm (base e) of <i>Number</i> .                                                           |
| Loop (normal)             | Perform a series of commands repeatedly: either the specified number of times or until <b>break</b> is encountered. |
| Loop (files & folders)    | Retrieves the specified files or folders, one at a time.                                                            |
| Loop (parse a string)     | Retrieves substrings (fields) from a string, one at a time.                                                         |
| Loop (read file contents) | Retrieves the lines in a text file, one at a time.                                                                  |
| Loop (registry)           | Retrieves the contents of the specified registry subkey, one item at a time.                                        |
| MAKELANGID                | Creates a <b>language identifier</b> from a primary language identifier and a sublanguage identifier.               |
| MAKELCID                  | Creates a <b>locale identifier</b> from a <b>language identifier</b> and a <b>sort order identifier</b> .           |
| MAKELONG                  | Creates a LONG value by concatenating the specified values.                                                         |
| MAKELPARAM                | Creates a value for use as an IParam parameter in a message.                                                        |
|                           | Creates a value for use as a return value from                                                                      |

|                  |                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAKELRESULT      | a window procedure.                                                                                                                                                       |
| MAKEWORD         | Creates a value for use as a return value from a window procedure.                                                                                                        |
| MAKEWPARAM       | Creates a value for use as an wParam parameter in a message.                                                                                                              |
| Menu             | Creates, deletes, modifies and displays menus and menu items. Changes the tray icon and its tooltip. Controls whether the main window of a compiled script can be opened. |
| MCODEH           | Creates a DynaCall object for machine code function.                                                                                                                      |
| MenuGetHandle    | Retrieves the Win32 menu handle of a menu.                                                                                                                                |
| MenuGetName      | Retrieves the name of a menu given a handle to its underlying Win32 menu.                                                                                                 |
| MenuSelect       | Invokes a menu item from the menu bar of the specified window.                                                                                                            |
| Mod              | Modulo. Returns the remainder when <i>Dividend</i> is divided by <i>Divisor</i> .                                                                                         |
| MonitorGet       | Retrieves screen resolution and multi-monitor info.                                                                                                                       |
| MouseClicked     | Clicks or holds down a mouse button, or turns the mouse wheel. NOTE: The Click command is generally more flexible and easier to use.                                      |
| MouseClickedDrag | Clicks and holds the specified mouse button, moves the mouse to the destination coordinates, then releases the button.                                                    |
| MouseGetPos      | Retrieves the current position of the mouse cursor, and optionally which window and control it is hovering over.                                                          |
| MouseMove        | Moves the mouse cursor.                                                                                                                                                   |

|                |                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MsgBox         | Displays the specified text in a small window containing one or more buttons (such as Yes and No).                                                            |
| NewThread      | Build-in function to start new exe thread (without using AutoHotkey.dll).                                                                                     |
| NumGet         | Returns the binary number stored at the specified address+offset.                                                                                             |
| NumPut         | Stores a number in binary format at the specified address+offset.                                                                                             |
| ObjAddRef      | Increments an object's reference count.                                                                                                                       |
| ObjByRef       | Creates an object containing <a href="#">ByRef</a> variables, obj.var returns actual variable content and obj.var := value will assign new value to variable. |
| ObjClone       | Creates a new object copying key and value items from given object (no deep copy is performed).                                                               |
| ObjGetAddress  | Returns the current address of the field's string buffer, if it has one.                                                                                      |
| ObjDelete      | Deletes key-value pairs from an object.                                                                                                                       |
| ObjDump        | Dump an object to memory or save to file for later use.                                                                                                       |
| ObjGetCapacity | Returns the current capacity of an object or one of its fields.                                                                                               |
| ObjGetAddress  | Returns the current address of the field's string buffer, if it has one.                                                                                      |
| ObjHasKey      | Returns true if Key is associated with a value (even "") within Object, otherwise false.                                                                      |
| ObjInsertAt    | Inserts values into an array.                                                                                                                                 |
| ObjLength      | Returns the array's length; that is, the highest integer key contained by the object, or 0 if                                                                 |

|                |                                                                                                                                       |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------|
|                | there aren't any.                                                                                                                     |
| ObjLoad        | Load a <b>dumped</b> object from memory or file.                                                                                      |
| ObjNewEnum     | Returns a new enumerator to enumerate this object's key-value pairs. This method is usually not called directly, but by the for-loop. |
| ObjPop         | Removes and returns the last array element.                                                                                           |
| ObjPush        | Appends values to the end of an array.                                                                                                |
| ObjRawSet      | Stores or overwrites a key-value pair in the object.                                                                                  |
| ObjRelease     | Decrements an object's reference count.                                                                                               |
| ObjRemoveAt    | Removes elements from an array.                                                                                                       |
| ObjSetCapacity | Adjusts the capacity of an object or one of its fields.                                                                               |
| ObjShare       | Create multi-thread save COM IDispatch proxy object.                                                                                  |
| OnExit         | Specifies a <b>function</b> to call automatically when the script exits.                                                              |
| OnMessage      | Specifies a <b>function</b> to call automatically when the script receives the specified message.                                     |
| Ord            | Returns the ordinal value (numeric character code) of the first character in the specified string.                                    |
| OutputDebug    | Sends a string to the debugger (if any) for display.                                                                                  |
| Pause          | Pauses the script's <b>current thread</b> .                                                                                           |
| PixelGetColor  | Retrieves the color of the pixel at the specified x,y coordinates.                                                                    |
| PixelSearch    | Searches a region of the screen for a pixel of                                                                                        |

|                    |                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------|
|                    | the specified color.                                                                                             |
| PostMessage        | Places a message in the message queue of a window or control.                                                    |
| ProcessClose       | Terminates a process.                                                                                            |
| Progress           | Creates or updates a window containing a progress bar.                                                           |
| ProcessExist       | Checks if a matching process exists (is running).                                                                |
| ProcessSetPriority | Changes the priority of a process.                                                                               |
| ProcessWait        | Waits until a matching process exists (is running).                                                              |
| ProcessWaitClose   | Waits for all matching processes to close.                                                                       |
| Random             | Generates a pseudo-random number.                                                                                |
| RegExMatch         | Determines whether a string contains a pattern (regular expression).                                             |
| RegExReplace       | Replaces occurrences of a pattern (regular expression) inside a string.                                          |
| RegDelete          | Deletes a value from the registry.                                                                               |
| RegDeleteKey       | Deletes a subkey from the registry.                                                                              |
| RegRead            | Reads a value from the registry.                                                                                 |
| RegWrite           | Writes a value to the registry.                                                                                  |
| RegisterCallback   | Creates a machine-code address that when called, redirects the call to a <a href="#">function</a> in the script. |
| Reload             | Replaces the currently running instance of the script with a new one.                                            |
| ResDelete          | Deletes a resource in executable file (dll or exe).                                                              |
|                    |                                                                                                                  |

|                                                                                                       |                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ResDllCreate                                                                                          | Creates a resource only dll. Such dll will not have any executable code but can be loaded into programs to access resources.                                                                                                                                              |
| ResExist                                                                                              | Check whether resource exists in the executable file (dll or exe).                                                                                                                                                                                                        |
| ResGet                                                                                                | Reads a resource from executable file (dll or exe).                                                                                                                                                                                                                       |
| ResPut                                                                                                | Updates a resource in executable file (dll or exe).                                                                                                                                                                                                                       |
| ResPutFile                                                                                            | Updates a resource in executable file (dll or exe).                                                                                                                                                                                                                       |
| Return                                                                                                | Returns from a subroutine to which execution had previously jumped via <a href="#">function-call</a> , <a href="#">Gosub</a> , <a href="#">Hotkey</a> activation, <a href="#">GroupActivate</a> , or other means.                                                         |
| Round                                                                                                 | If <i>N</i> is omitted or 0, <i>Number</i> is rounded to the nearest integer. If <i>N</i> is positive number, <i>Number</i> is rounded to <i>N</i> decimal places. If <i>N</i> is negative, <i>Number</i> is rounded by <i>N</i> digits to the left of the decimal point. |
| Run                                                                                                   | Runs an external program.                                                                                                                                                                                                                                                 |
| RunAs                                                                                                 | Specifies a set of user credentials to use for all subsequent uses of <a href="#">Run</a> and <a href="#">RunWait</a> .                                                                                                                                                   |
| RunWait                                                                                               | Runs an external program and waits until it finishes.                                                                                                                                                                                                                     |
| <a href="#">Send</a> / <a href="#">SendRaw</a> / <a href="#">SendInput</a> / <a href="#">SendPlay</a> | Sends simulated keystrokes and mouse clicks to the <a href="#">active</a> window.                                                                                                                                                                                         |
| SendLevel                                                                                             | Controls which artificial keyboard and mouse events are ignored by hotkeys and hotstrings.                                                                                                                                                                                |

|                      |                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SendMessage          | Sends a message to a window or control and waits for acknowledgement.                                                                                                                                                                     |
| SendMode             | Makes <code>Send</code> synonymous with <code>SendInput</code> or <code>SendPlay</code> rather than the default ( <code>SendEvent</code> ). Also makes <code>Click</code> and <code>MouseMove/Click/Drag</code> use the specified method. |
| SetCapslockState     | Sets the state of the Capslock key. Can also force the key to stay on or off.                                                                                                                                                             |
| SetControlDelay      | Sets the delay that will occur after each control-modifying command.                                                                                                                                                                      |
| SetDefaultMouseSpeed | Sets the mouse speed that will be used if unspecified in <code>Click</code> and <code>MouseMove/Click/Drag</code> .                                                                                                                       |
| SetKeyDelay          | Sets the delay that will occur after each keystroke sent by <code>Send</code> or <code>ControlSend</code> .                                                                                                                               |
| SetMouseDelay        | Sets the delay that will occur after each mouse movement or click.                                                                                                                                                                        |
| SetNumlockState      | Sets the state of the Numlock key. Can also force the key to stay on or off.                                                                                                                                                              |
| SetScrollLockState   | Sets the state of the Scrolllock key. Can also force the key to stay on or off.                                                                                                                                                           |
| SetRegView           | Allows registry commands in a 32-bit script to access the 64-bit registry view and vice versa.                                                                                                                                            |
| SetStoreCapslockMode | Whether to restore the state of CapsLock after a <code>Send</code> .                                                                                                                                                                      |
| SetTimer             | Causes a subroutine to be launched automatically and repeatedly at a specified time interval.                                                                                                                                             |
|                      | Sets the matching behavior of the WinTitle                                                                                                                                                                                                |

|                          |                                                                                                              |
|--------------------------|--------------------------------------------------------------------------------------------------------------|
| <b>SetTitleMatchMode</b> | parameter in commands such as <a href="#">WinWait</a> .                                                      |
| <b>SetWinDelay</b>       | Sets the delay that will occur after each windowing command, such as <a href="#">WinActivate</a> .           |
| <b>SetWorkingDir</b>     | Changes the script's current working directory.                                                              |
| <b>Shutdown</b>          | Shuts down, restarts, or logs off the system.                                                                |
| <b>Sin</b>               | Returns the trigonometric sine of <i>Number</i> .                                                            |
| <b>Sleep</b>             | Waits the specified amount of time before continuing.                                                        |
| <b>Sort</b>              | Arranges a variable's contents in alphabetical, numerical, or random order (optionally removing duplicates). |
| <b>SoundBeep</b>         | Emits a tone from the PC speaker.                                                                            |
| <b>SoundGet</b>          | Retrieves various settings from a sound device (master mute, master volume, etc.)                            |
| <b>SoundPlay</b>         | Plays a sound, video, or other supported file type.                                                          |
| <b>SoundSet</b>          | Changes various settings of a sound device (master mute, master volume, etc.)                                |
| <b>SplitPath</b>         | Separates a file name or URL into its name, directory, extension, and drive.                                 |
| <b>Sqrt</b>              | Returns the square root of <i>Number</i> .                                                                   |
| <b>SplashImage</b>       | Creates or updates a window containing an image.                                                             |
| <b>SplashTextOn</b>      | Creates a customizable text popup window.                                                                    |
| <b>StatusBarGetText</b>  | Retrieves the text from a standard status bar control.                                                       |
| <b>StatusBarWait</b>     | Waits until a window's status bar contains the specified string.                                             |

|                 |                                                                                                                                 |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------|
| StrGet          | Reads a string from a memory address, optional converting it between code pages.                                                |
| StrPut          | Copies a string to a memory address, optional converting it between code pages.                                                 |
| StrPutVar       | Copies a string to a variable, optionally converting to or from a given code page.                                              |
| StringCaseSense | Determines whether string comparisons are case sensitive (default is "not case sensitive").                                     |
| StrLen          | Retrieves the count of how many characters are in a string.                                                                     |
| StrLower        | Converts a string to lowercase.                                                                                                 |
| StrReplace      | Replaces the specified substring with a new string.                                                                             |
| StrSplit        | Separates a string into an array of substrings using the specified delimiters.                                                  |
| Struct          | Struct is a build-in function that creates and returns a special structure object.                                              |
| StrUpper        | Converts a string to uppercase.                                                                                                 |
| SubStr          | Retrieves one or more characters from the specified position in a string.                                                       |
| Suspend         | Disables or enables all or selected <a href="#">hotkeys</a> and <a href="#">hotstrings</a> .                                    |
| SysGet          | Retrieves screen resolution, multi-monitor info, dimensions of system objects, and other system properties.                     |
| Tan             | Returns the trigonometric tangent of <i>Number</i> .                                                                            |
| Thread          | Sets the priority or interruptibility of <a href="#">threads</a> . It can also temporarily disable all <a href="#">timers</a> . |
| ThreadObj       | Included function to start new exe thread (without using AutoHotkey.dll).                                                       |

|                   |                                                                                                                                         |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Throw             | Signals the occurrence of an error. This signal can be caught by a <code>try-catch</code> statement.                                    |
| ToChar            | Convert an integer to signed char type.                                                                                                 |
| ToInt             | Convert an integer to signed int type.                                                                                                  |
| ToolTip           | Creates an always-on-top window anywhere on the screen.                                                                                 |
| ToShort           | Convert an integer to signed short type.                                                                                                |
| ToUChar           | Convert an integer to unsigned char type.                                                                                               |
| ToUInt            | Convert an integer to unsigned int type.                                                                                                |
| ToUShort          | Convert an integer to unsigned short type.                                                                                              |
| TrayTip           | Creates a balloon message window or toast notification near the tray icon.                                                              |
| Trim              | Trims certain characters from the beginning and/or end of a string.                                                                     |
| Try               | Guards one or more statements (commands or expressions) against runtime errors and exceptions thrown by the <code>throw</code> command. |
| Type              | Returns the exact type of a value.                                                                                                      |
| Until             | Applies a condition to the continuation of a Loop or For-loop.                                                                          |
| UnZip             | This function is used to unzip a zip archive to disk.                                                                                   |
| UnZipBuffer       | This function is used to unzip an item from zip archive to memory,.                                                                     |
| UnZipRawMemory    | This function is used to unzip and decrypt raw memory created with ZipRawMemory, for example from resource.                             |
| Var := expression | Evaluates an <code>expression</code> and stores the result in a <code>variable</code> .                                                 |

|                    |                                                                                                                                                          |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| VarSetCapacity     | Enlarges a variable's holding capacity or frees its memory. Normally, this is necessary only for unusual circumstances such as <a href="#">DllCall</a> . |
| While-loop         | Performs a series of commands repeatedly until the specified <a href="#">expression</a> evaluates to false.                                              |
| WinActivate        | Activates the specified window (makes it foremost).                                                                                                      |
| WinActivateBottom  | Same as <a href="#">WinActivate</a> except that it activates the bottommost (least recently active) matching window rather than the topmost.             |
| WinActive          | Returns the Unique ID (HWND) of the active window if it matches the specified criteria.                                                                  |
| WinClose           | Closes the specified window.                                                                                                                             |
| WinExist           | Returns the Unique ID (HWND) of the first matching window.                                                                                               |
| WinGetClass        | Retrieves the specified window's class name.                                                                                                             |
| WinGetControls     | Retrieves an array of control names for all controls in the specified window.                                                                            |
| WinGetControlsHwnd | Retrieves an array of control unique ID numbers (HWND/handle) for all controls in the specified window.                                                  |
| WinGetCount        | Retrieves the count of matching windows.                                                                                                                 |
| WinGetExStyle      | Retrieves an integer representing the extended style of the specified window.                                                                            |
| WinGetID           | Retrieves the unique ID number (HWND/handle) of a window.                                                                                                |
| WinGetIDLast       | Retrieves the unique ID number (HWND/handle) of the last/bottommost window.                                                                              |

|                   |                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------|
| WinGetList        | Retrieves the unique ID numbers of all matching windows.                                          |
| WinGetMinMax      | Retrieves the minimized/maximized state for a window.                                             |
| WinGetPID         | Retrieves the <a href="#">Process ID (PID)</a> of a window.                                       |
| WinGetPIDList     | Retrieves the <a href="#">Process IDs (PID)</a> of all matching windows.                          |
| WinGetPos         | Retrieves the position and size of the specified window.                                          |
| WinGetProcessName | Retrieves the name of the process (e.g. notepad.exe) that owns a window.                          |
| WinGetProcessPath | Retrieves the full path and name of the process (e.g. C:\Windows\notepad.exe) that owns a window. |
| WinGetStyle       | Retrieves an 8-digit hexadecimal number representing the style (respectively) of a window.        |
| WinGetText        | Retrieves the text from the specified window.                                                     |
| WinGetTitle       | Retrieves the title of the specified window.                                                      |
| WinGetTransColor  | Retrieves the color that is marked transparent in a window.                                       |
| WinGetTransparent | Retrieves the degree of transparency of a window.                                                 |
| WinHide           | Hides the specified window.                                                                       |
| WinKill           | Forces the specified window to close.                                                             |
| WinMaximize       | Enlarges the specified window to its maximum size.                                                |
|                   | Collapses the specified window into a button                                                      |

|                    |                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------|
| WinMinimize        | on the task bar.                                                                                                            |
| WinMinimizeAll     | Minimizes all windows.                                                                                                      |
| WinMinimizeAllUndo | Reverses the effect of a previous WinMinimizeAll.                                                                           |
| WinMove            | Changes the position and/or size of the specified window.                                                                   |
| WinRedraw          | Attempts to update the appearance/contents of a window by informing the OS that the window's rectangle needs to be redrawn. |
| WinRestore         | Unminimizes or unmaximizes the specified window if it is minimized or maximized.                                            |
| WinSetAlwaysOnTop  | Makes a window stay on top of all other windows.                                                                            |
| WinSetEnabled      | Disables or enables a window (respectively).                                                                                |
| WinSetExStyle      | Changes the extended style of a window, respectively.                                                                       |
| WinSetRegion       | Changes the shape of a window to be the specified rectangle, ellipse, or polygon.                                           |
| WinSetStyle        | Changes the style of a window, respectively.                                                                                |
| WinSetTitle        | Changes the title of the specified window.                                                                                  |
| WinSetTransColor   | Makes all pixels of the chosen color invisible inside the target window.                                                    |
| WinSetTransparent  | Makes a window semi-transparent.                                                                                            |
| WinShow            | Unhides the specified window.                                                                                               |
| WinWait            | Waits until the specified window exists.                                                                                    |
| WinWaitActive      | Waits until the specified window is active.                                                                                 |
|                    |                                                                                                                             |

|                                |                                                                                                                                            |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <code>WinWaitClose</code>      | Waits until the specified window does not exist.                                                                                           |
| <code>WinWaitNotActive</code>  | Waits until the specified window is not active.                                                                                            |
| <code>ZipAddBuffer</code>      | Add a file to zip archive from memory.                                                                                                     |
| <code>ZipAddFile</code>        | Add a file to zip archive from disk.                                                                                                       |
| <code>ZipAddFolder</code>      | Add an empty folder to zip archive.                                                                                                        |
| <code>ZipCloseBuffer</code>    | Close zip archive created with <code>ZipCreateBuffer</code> .                                                                              |
| <code>ZipCloseFile</code>      | Close zip archive created with <code>ZipCreateFile</code> .                                                                                |
| <code>ZipCreateBuffer</code>   | Create a zip archive in memory.                                                                                                            |
| <code>ZipCreateFile</code>     | Create a zip archive on disk.                                                                                                              |
| <code>ZipInfo</code>           | Get information about all items in zip archive.                                                                                            |
| <code>ZipOptions</code>        | Change options for a zip archive created with <code>ZipCreateFile</code> or <code>ZipCreateBuffer</code> .                                 |
| <code>ZipRawMemory</code>      | Create a raw zip archive in memory, <code>UnZipRawMemory</code> must be used to unzip such an archive.                                     |
| <code>#ClipboardTimeout</code> | Changes how long the script keeps trying to access the clipboard when the first attempt fails.                                             |
| <code>#DllImport</code>        | Creates a script function with predefined parameters to call a dll function.                                                               |
| <code>#ErrorStdOut</code>      | Sends any syntax error that prevents a script from launching to stdout rather than displaying a dialog.                                    |
| <code>#HotkeyInterval</code>   | Along with <code>#MaxHotkeysPerInterval</code> , specifies the rate of hotkey activations beyond which a warning dialog will be displayed. |
|                                |                                                                                                                                            |

|                               |                                                                                                                                                                                                       |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #HotkeyModifierTimeout        | Affects the behavior of <a href="#">hotkey</a> modifiers: CTRL, ALT, WIN, and SHIFT.                                                                                                                  |
| #Hotstring                    | Changes <a href="#">hotstring</a> options or ending characters.                                                                                                                                       |
| #If                           | Similar to <a href="#">#IfWinActive</a> , but for arbitrary expressions.                                                                                                                              |
| #IfTimeout                    | Sets the maximum time that may be spent evaluating a single <a href="#">#If</a> expression.                                                                                                           |
| #IfWinActive /<br>#IfWinExist | Creates context-sensitive <a href="#">hotkeys</a> and <a href="#">hotstrings</a> . Such hotkeys perform a different action (or none at all) depending on the type of window that is active or exists. |
| #Include                      | Causes the script to behave as though the specified file's contents are present at this exact position.                                                                                               |
| #InputLevel                   | Controls which artificial keyboard and mouse events are ignored by hotkeys and hotstrings.                                                                                                            |
| #InstallKeybdHook             | Forces the unconditional installation of the keyboard hook.                                                                                                                                           |
| #InstallMouseHook             | Forces the unconditional installation of the mouse hook.                                                                                                                                              |
| #KeyHistory                   | Sets the maximum number of keyboard and mouse events displayed by the <a href="#">KeyHistory</a> window. You can set it to 0 to disable key history.                                                  |
| #MaxHotkeysPerInterval        | Along with <a href="#">#HotkeyInterval</a> , specifies the rate of <a href="#">hotkey</a> activations beyond which a warning dialog will be displayed.                                                |
| #MaxThreads                   | Sets the maximum number of simultaneous <a href="#">threads</a> .                                                                                                                                     |
|                               | Causes some or all <a href="#">hotkeys</a> to buffer rather than ignore keypresses when their                                                                                                         |

|                                   |                                                                                                                |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>#MaxThreadsBuffer</code>    | <code>#MaxThreadsPerHotkey</code> limit has been reached.                                                      |
| <code>#MaxThreadsPerHotkey</code> | Sets the maximum number of simultaneous threads per hotkey or hotstring.                                       |
| <code>#MenuMaskKey</code>         | Changes which key is used to mask Win or Alt keyup events.                                                     |
| <code>#NoTrayIcon</code>          | Disables the showing of a tray icon.                                                                           |
| <code>#Persistent</code>          | Keeps a script permanently running (that is, until the user closes it or <code>ExitApp</code> is encountered). |
| <code>#SingleInstance</code>      | Determines whether a script is allowed to run again when it is already running.                                |
| <code>#UseHook</code>             | Forces the use of the hook to implement all or some keyboard hotkeys.                                          |
| <code>#Warn</code>                | Enables or disables warnings for selected conditions that may be indicative of developer errors.               |
| <code>#WinActivateForce</code>    | Skips the gentle method of activating a window and goes straight to the forceful method.                       |

# AutoHotkey.dll module

## Table of Contents

- [Why do we need an AutoHotkey Module](#)
- [Using AutoHotkey Module in other programs](#)
- [COM Interface](#)
- [Reload script but keep Process running.](#)

## Why do we need an AutoHotkey Module

**Multi-threading:** AutoHotkey is not designed for multi-threading naturally and most likely it will be never implemented. Using AutoHotkey.dll we can still run multiple scripts in one process, on a multi-core system we can even run multiple scripts at the same time. To do so, AutoHotkey module also needs to be loaded multiple times. Note for this you will need to use [MemoryModule](#) or multiple dlls by copying and renaming AutoHotkey.dll for example to MyScript1.dll, MyScript2.dll ...

AutoHotkey Module is running in its own context, using separate memory, functions and variables, this allows to access, read and write its memory / variables from another thread. AutoHotkey Module exports several functions that allow various operations to create and manipulate the new thread.

**Use AutoHotkey in other programs:** AutoHotkey Module can be also used in other programming languages like VB, C#, C++, Python, Lua and many more. Programs that do not support loading a dll naturally can use the [COM Interface](#) of AutoHotkey.dll.

**AutoHotkey COM Interface:** AutoHotkey.dll can be also loaded and manipulated using its [COM Interface](#). Before using this Interface, AutoHotkey.dll needs to be registered:

```
regsvr32 "C:\Program
Files\AutoHotkey\AutoHotkey.dll"
```

```
regsvr32 /u "C:\Program
Files\AutoHotkey\AutoHotkey.dll"
```

AutoHotkey\_H additionally supports loading unregistered dlls using [ComObjDll](#).

Internally COM Interface always use [MemoryModule](#) to create a new thread, this allows loading the same module multiple times. AutoHotkey.dll automatically frees the module when COM object is released.

### **Reload script but keepProcess running**

Using AutoHotkey Module we can share variables/objects/memory between thread. For example we can create all required variables in main thread and share them to AutoHotkey Module using [Alias](#) function. Whenever a reload is required (e.g to accept new hotstrings and hotkeys) we can reload the AutoHotkey module only and share the variables again.

# AutoHotkey COM Interface

## Available Interfaces

| Module AutoHotkey v1               | Interface           | GUID                                   |
|------------------------------------|---------------------|----------------------------------------|
| Last registered (Unicode/ANSI/X64) | AutoHotkey.Script   | {C00BCC8C-5A04-4392-870F-20AAE1B926B2} |
| Unicode 32-bit                     | AutoHotkey.UNICODE  | {C58DCD96-1D6F-4F85-B555-02B7F21F5CAF} |
| ANSI 32-bit                        | AutoHotkey.ANSI     | {974318D9-A5B2-4FE5-8AC4-33A0C9EBB8B5} |
| Unicode 64-bit                     | AutoHotkey.X64      | {38D00012-DC83-4E17-9BAD-D9DD97902580} |
| Module AutoHotkey v2               | Interface           | GUID                                   |
| Last registered (Unicode/X64)      | AutoHotkey2.Script  | {FEEEC4BA-04AF-45F0-B385-7290C65CFB9B} |
| Unicode win32                      | AutoHotkey2.UNICODE | {EC81EBBA-6CEE-4363-AB77-C0E57046AA89} |
| Unicode x64                        | AutoHotkey2.X64     | {F1D0DE03-30FD-4326-B33F-989BBFAA5FFA} |

## Examples

### Visual Basic example.

```
Sub atest()
Dim AhkCom As Object
Set AhkCom = CreateObject("AutoHotkey.Script")
AhkCom.ahktextdll("MsgBox Hello World!" &
Chr(13) & "ExitApp")
End Sub
```

### AutoHotkey example.

```
AhkCom := ComObjCreate("AutoHotkey.Script")
AhkCom.ahktextdll("MsgBox Hello
World!`nExitApp")
While AhkCom.ahkReady()
Sleep, 100
MsgBox Exiting now
```

### ComObjDll example

```
hModule:=DllCall("LoadLibrary","Str",A_AhkDir
"\AutoHotkey.dll")
ahk:=ComObjDll(hModule,"{FEEEC4BA-04AF-45F0-
B385-7290C65CFB9B}") ; CLSID for Version 2.0

hModule:=MemoryLoadLibrary(A_AhkDir
"\AutoHotkey.dll")
ahk:=ComObjMemDll(hModule,"{FEEEC4BA-04AF-45F0-
B385-7290C65CFB9B}") ; CLSID for Version 2.0
```

# ahkdll

Exported function to create additional AutoHotkey thread in current process from a file on disk or network.

If a thread is already executed it will be terminated before the new thread starts.

Note you will need to load AutoHotkey.dll module using LoadLibrary before you can use this function, see Example.

**This function is only available in AutoHotkey.dll**

```
OutputVar := DllCall("AutoHotkey.dll\ahkdll", "Str",
"FileName", "Str", "Parameters", "Str", "Title",
"CDecl UPTR")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkdll", "Str", "FileName.ahk",
"Str",, "Str",, "CDecl UPTR"
Function Example: hThread :=
DllCall("AutoHotkey.dll\ahkdll", "Str",
"FileName.ahk", "Str",, "Str",, "PTR")
```

## Parameters

### OutputVar

The name of the variable in which to store the handle of the newly created thread.

### FileName.ahk (optional)

New AutoHotkey script saved on disk or network to be executed in AutoHotkey.dll. When omitted, following script is used:

```
#Persistent
#NoTrayIcon
```

### Parameters (optional)

Command line parameters that will be available in built-in variable `A_Args` object.

### Title (optional)

Title for the dll thread (by default filename) that will be shown in MsgBox, Gui... when no Title is given.

## General Remarks

ahkdll behaves different when instead of MyScript.ahk an empty string or 0 is passed:

- If AutoHotkey.dll is compiled, the compiled script is executed.
- Otherwise an empty, persistent Script will be started.

When exe was started with command line parameters, the dll will be able to grab the parameters and use same library as well as `A_ScriptDir` and `A_ScriptFullPath` as the executable. When executable is compiled, the path of the executable will be used.

## Related

[ahktextdll](#), [AutoHotkey.dll](#), [AhkThread](#), [Threads](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath "\ahkdll","Str",A_ScriptDir
"\MyDllScript.ahk","Str","", "CDecl") ; start a new
thread from file.
While DllCall(dllpath "\ahkReady")
 Sleep 100 ; wait for the thread to exit

; Same Example like above using included
AutoHotkey.dll
dll:=AhkThread(A_ScriptDir
"\MyDllScript.ahk","",true)
While dll.ahkReady()
 Sleep 100 ; wait for the thread to exit
```

# ahktextdll

Exported function to create additional AutoHotkey thread in current process from a file on disk or network.

If a thread is already executed it will be terminated before the new thread starts.

Note you will need to load AutoHotkey.dll module using LoadLibrary before you can use this function, see Example.

**This function is only available in AutoHotkey.dll**

```
OutputVar := DllCall("AutoHotkey.dll\ahktextdll",
"Str", "Script", "Str", "Parameters", "Str", "Title",
"CDecl UPTR")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahktextdll", "Str", "MsgBox
From Thread", "Str",,, "Str",,, "CDecl UPTR"
Function Example: hThread :=
DllCall("AutoHotkey.dll\ahktextdll", "Str",
"MsgBox From Thread", "Str",,, "Str",,, "PTR")
```

## Parameters

### OutputVar

The name of the variable in which to store the handle of the newly created thread.

### Script (optional)

String containing new AutoHotkey script to be executed in AutoHotkey.dll. When omitted, following script is used:

```
#Persistent
#NoTrayIcon
```

### Parameters (optional)

Command line parameters that will be available in built-in variable `A_Args` object.

### Title (optional)

Title for the dll thread (by default filename) that will be shown in MsgBox, Gui... when no Title is given.

## General Remarks

ahkdll behaves different when instead of MyScript.ahk an empty string or 0 is passed:

- If AutoHotkey.dll is compiled, the compiled script is executed.
- Otherwise an empty, persistent Script will be started.

When exe was started with command line parameters, the dll will be able to grab the parameters and use same library as well as `A_ScriptDir` and `A_ScriptFullPath` as the executable. When executable is compiled, the

path of the executable will be used.

## Related

[Threads](#) [DllCall](#)

[ahktextdll](#), [AutoHotkey.dll](#), [AhkThread](#),

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath "\ahktextdll","Str","MsgBox Hello
World!","Str","","CDecl") ; start a new thread
from file.
While DllCall(dllpath "\ahkReady")
 Sleep 100 ; wait for the thread to exit

; Same Example like above using included
AutoHotkey.dll
dll:=AhkThread("MsgBox Hello World!")
While dll.ahkReady()
 Sleep 100 ; wait for the thread to exit
```

# AhkThread

Create a real additional AutoHotkey thread in current process using AutoHotkey.dll.

```
OutputVar := AhkThread (ScriptOrFile, Parameters,
Title, ScriptIsFile, DllToUse)
```

```
Function Example: Thread := AhkThread("MsgBox
Message from thread.")
```

## Parameters

### OutputVar

The name of the variable in which to store the object for newly created thread.

Using this object we can call all AutoHotkey.dll functions with simple object syntax.

### ScriptOrFile (optional)

The AutoHotkey script to execute. This parameter can be one of the following:

- Script passed as string or variable containing string.
- Path to an ahk file. Parameter ScriptIsFile must be set to true to start a

```
script from file.
- Empty or omitted to start an empty script
("#Persistent`n#NoTrayIcon").
- 0 to prepare the thread but not start it.
Use ahkdll or ahktextdll to start new
thread.
```

### Parameters (optional)

Command line parameters that will be available in built-in variable `A_Args` object.

### Title (optional)

Title for the dll thread (by default filename) that will be shown in MsgBox, Gui... when no Title is given.

### ScriptIsFile (optional)

Set to true or 1 if ScriptOrFile is a path to a file on disk or network.

### DllToUse (optional)

Path or Resource name for AutoHotkey.dll that will be used to create new thread.

## General Remarks

ahkdll and ahktextdll will behave different when AutoHotkey.dll is compiled.

When ScriptOrFile is omitted or empty, the compiled script is executed.

To free resources for the thread we have to call `ahkthread_free(obj)` and

also release the object (obj:= "").

obj.ahkterminate() can be called optionally.

## Methods

|                                 |                                                                                                                                                      |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ahkdll</a>          | Load a new thread from a file, current thread will be terminated.                                                                                    |
| <a href="#">ahktextdll</a>      | Load a new thread from a string/memory/variable, current thread will be terminated.                                                                  |
| <a href="#">ahkReady</a>        | Returns 1 (true) if a thread is being executed currently, 0 (false) otherwise.                                                                       |
| <a href="#">ahkTerminate</a>    | Terminate thread.                                                                                                                                    |
| <a href="#">ahkReload</a>       | Reload thread using same parameters used with <a href="#">ahkdll</a> or <a href="#">ahktextdll</a> .                                                 |
| <a href="#">ahkFunction</a>     | Call a function via SendMessage method. Mainly used with AutoHotkey.dll to call a function in dll script or call a function in main script from dll. |
| <a href="#">ahkPostFunction</a> | Call a function via PostMessage method (does not wait until function returns). Also used mainly with AutoHotkey.dll                                  |
| <a href="#">ahkExecuteLine</a>  | Executes script from given line pointer.                                                                                                             |
| <a href="#">ahkLabel</a>        | Goto (PostMessage) or Gosub (SendMessage) a Label. Also used mainly with AutoHotkey.dll                                                              |
| <a href="#">ahkFindFunction</a> | Find a function and return its pointer.                                                                                                              |
| <a href="#">ahkFindLabel</a>    | Find a label and return its pointer.                                                                                                                 |
| <a href="#">addFile</a>         | Add and optionally execute additional script/code from file. Not available for scripts compiled with AutoHotkeySC.bin.                               |
| <a href="#">addScript</a>       | Add and optionally execute additional script/code from text/memory/variable. Not available for scripts compiled with AutoHotkeySC.bin.               |
| <a href="#">ahkExec</a>         | Execute some script/code from text/memory/variable temporarily. Not available for scripts compiled with AutoHotkeySC.bin.                            |
| <a href="#">ahkassign</a>       | Assign a value to variable or pointer of variable.                                                                                                   |

|           |                                   |
|-----------|-----------------------------------|
| ahkgetvar | Retrieve a value from a variable. |
| ahkPause  | Pause Script.                     |

## Related

[AutoHotkey.dll](#), [Objects](#), [DllCall](#)

## Examples

```
ahkdll:=AhkThread("Msgbox `%
variable:=`"Thread`") ; Loads the AutoHotkey
module and starts the script.
While !ahkdll.ahkgetvar.variable
 Sleep 50 ; wait until variable has been set.
MsgBox % ahkdll.ahkgetvar.variable ; Display
content of variable in thread
ahkthread_free(ahkdll),ahkdll:="" ; Stop execution
in thread and free resources.
```

# AhkExported

Retrieves an object to control main thread (AutoHotkey.exe) same way as in [AhkThread](#).

```
OutputVar := AhkExported()
```

```
Function Example: AhkExe := AhkExported()
```

## Parameters

### OutputVar

The name of the variable in which to store the AutoHotkey.exe object. Using this object we can call all AutoHotkey.exe functions using object syntax.

## General Remarks

Scripts compiled with AutoHotkeySC.bin cannot use ahkExec, addFile and addScript functions.

### Methods

- [ahkFunction](#) Call a function via SendMessage method.
- [ahkPostFunction](#) Call a function via PostMessage method (does not wait until function returns).
- [ahkExecuteLine](#) Executes script from given line pointer.
- [ahkLabel](#) Goto (PostMessage) or Gosub (SendMessage) a Label.
- [ahkFindFunction](#) Find a function and return its pointer.

|                           |                                                                              |
|---------------------------|------------------------------------------------------------------------------|
| <code>ahkFindLabel</code> | Find a label and return its pointer.                                         |
| <code>addFile</code>      | Add and optionally execute additional script/code from file.                 |
| <code>addScript</code>    | Add and optionally execute additional script/code from text/memory/variable. |
| <code>ahkExec</code>      | Execute some script/code from text/memory/variable temporarily.              |
| <code>ahkassign</code>    | Assign a value to variable or pointer of variable.                           |
| <code>ahkgetvar</code>    | Retrieve a value from a variable.                                            |
| <code>ahkPause</code>     | Pause Script.                                                                |

## Related

[AutoHotkey.dll](#), [AhkThread](#), [Objects](#), [DllCall](#)

## Examples

```
AhkExe:=AhkExported()
Msgbox % AhkExe.ahkgetvar("A_IsCompiled") ; Check
if Script is compiled e.g. from AutoHotkey.dll
script
```

# ahkgetvar

Exported function to retrieve the value of a variable as string.

```
OutputVar := DllCall("Module\ahkgetvar", "Str",
"VarName", "UInt", GetVarPointer, "CDecl Str")
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkgetvar", "Str",
"MyVar", "UInt", 0, "CDecl Str")
Result :=
DllCall("AutoHotkey.exe\ahkgetvar", "Str",
"MyVar", "UInt", 0, "CDecl Str")
```

## Parameters

### OutputVar

The name of variable in which to store the value of variable as string or variable pointer if GetvarPointer is 1 / true.

### VarName

The name of the variable to get pointer or contents from.

### GetVarPointer

1 / true to receive a pointer to variable, FALSE / NULL / 0 to receive content of variable, similar to [GetVar](#).

## Related

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "", "Str", "", "CDecl") ; start a
new thread from file.
DllCall(dllpath
"\ahkassign", "Str", "var", "Str", "value", "CDecl") ;
assing value to var
MsgBox % DllCall(dllpath
"\ahkgetvar", "Str", "var", "UInt", 0, "CDecl") ; wait
for the thread to exit

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread()
dll.ahkassign("var", "value")
MsgBox % dll.ahkgetvar.var
```

# ahkassign

Exported function that assigns a string value to a variable in currently executed script.

```
OutputVar := DllCall("Module\ahkassign", "Str",
"VarName", "Str", Value, "CDecl UInt")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkassign", "Str", "Variable",
"Str", "100", "CDecl UInt"
DllCall
"AutoHotkey.exe\ahkassign", "Str", "Variable",
"Str", "100", "CDecl UInt"
```

```
Function Example: Result :=
DllCall("AutoHotkey.dll\ahkassign", "Str",
"Variable", "Str", "100", "CDecl UInt")
Result :=
DllCall("AutoHotkey.exe\ahkassign", "Str",
"Variable", "Str", "100", "CDecl UInt")
```

## Parameters

### OutputVar

The name of the variable in which to store the result, -1 on failure and 0 on success.

### VarName

Name of the variable to assign value to.

Value

Value as String.

## Related

[ahkFindFunc](#), [ahkFindLabel](#), [ahkgetvar](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "", "Str", "", "CDecl") ; start a
new thread from file.
DllCall(dllpath
"\ahkassign", "Str", "var", "Str", "value", "CDecl") ;
assing value to var
MsgBox % DllCall(dllpath
"\ahkgetvar", "Str", "var", "UInt", 0, "CDecl") ; wait
for the thread to exit

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread()
dll.ahkassign("var", "value")
MsgBox % dll.ahkgetvar.var
```

# ahkLabel

Exported function that allows calling a Label in script currently executed by AutoHotkey module.

Parameters can be strings only as well as return value.

```
OutputVar := DllCall("Module\ahkLabel", "Str",
"LabelName", "UInt", DoNotWait, "CDecl Int")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkLabel", "Str", "MyLabel",
"UInt", 0, "CDecl Int"
DllCall
"AutoHotkey.exe\ahkLabel", "Str", "MyLabel",
"UInt", 0, "CDecl Int")
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkLabel", "Str",
"MyLabel", "UInt", 0, "CDecl Int")
OutputVar :=
DllCall("AutoHotkey.exe\ahkLabel", "Str",
"MyLabel", "UInt", 0, "CDecl Int")
```

## Parameters

### OutputVar

The name of the variable to store 1 / true if label was found or 0 / false otherwise.

## LabelName

The name of the label to jump to.

## DoNotWait

1 (true) will not wait for the code to finish / return, FALSE / NULL / 0 will wait for execution to finish like GoSub does.

## Related

[ahkPostFunction](#), [ahkFindFunc](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath "\ahktextdll","Str",
(
#Persistent
MyLabel:
MsgBox `% A_ThisLabel
Return
),"Str","", "CDecl") ; start a new thread, just
the function.
DllCall(dllpath
"\ahkLabel","Str","MyLabel","UInt",0,"CDecl") ;
jump to label and wait for it to finish / return.

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread("
(
```



# ahkFunction

Exported function that allows calling a function in script currently executed by AutoHotkey module.

Parameters can be strings only as well as return value.

```
OutputVar := ahkFunction("FuncName" [, Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10])
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkFunction", "Str", "FuncName"
[, "Str", "Arg1", ..., "Str", "Arg10"], "cdecl
Str"

DllCall
"AutoHotkey.exe\ahkFunction", "Str", "FuncName"
[, "Str", "Arg1", ..., "Str", "Arg10"], "cdecl
Str"
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkFunction", "Str",
"FuncName" [, "Str", Arg1 , ..., "Str", Arg10],
"Str")

OutputVar :=
DllCall("AutoHotkey.exe\ahkFunction", "Str",
"FuncName" [, "Str", Arg1 , ..., "Str", Arg10],
"Str")
```

## Parameters

## OutputVar

The name of the variable to store the functions returned value as string, on failure empty string is stored.

## FuncName

The name of the function to call.

## Arg1, ..., Arg10

You can pass up to 10 parameters to the function. Note all parameters need to be Strings or NULL / 0 if you want to omit parameter (use function default parameters)

## Remarks

Note that Function will not run if [#MaxThreads](#) limit is reached.

## Related

[ahkPostFunction](#), [ahkFindFunc](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "#Persistent`nMyFunc(param)
{`nSleep 10000`nMsgBox `%
param`n}", "Str", "", "CDecl") ; start a new thread,
```

just the function.

```
Msgbox % DllCall(dllpath
"\ahkFunction", "Str", "MyFunc", "Str", "Hello
World!", "CDecl Str") ; call the function.
```

; Same example like above using internal  
AutoHotkey.dll

```
dll:=AhkThread("#Persistent`nMyFunc(param){`nSleep
10000`nMsgBox `% param`n}")
MsgBox % dll.ahkFunction["MyFunc", "test"]
```

# ahkPostFunction

Exported function that allows calling a function in script currently executed by AutoHotkey module without waiting for the function to return.

Parameters can be strings only.

```
OutputVar := ahkPostFunction("FuncName" [, Arg1,
Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9,
Arg10])
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkPostFunction", "Str",
"FuncName" [, "Str", "Arg1", ..., "Str",
"Arg10"], "CDecl Str"

DllCall
"AutoHotkey.exe\ahkPostFunction", "Str",
"FuncName" [, "Str", "Arg1", ..., "Str",
"Arg10"], "CDecl Str"
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkPostFunction",
"Str", "FuncName" [, "Str", Arg1 , ...,
"Str", Arg10], "Str")

OutputVar :=
DllCall("AutoHotkey.exe\ahkPostFunction",
"Str", "FuncName" [, "Str", Arg1 , ...,
"Str", Arg10], "Str")
```

## Parameters

## OutputVar

The name of the variable to store 0 if function was found or -1 if function was not found.

## FuncName

The name of the function to call.

## Arg1, ..., Arg10

You can pass up to 10 parameters to the function. Note all parameters need to be Strings or NULL / 0 if you want to omit parameter (use function default parameters)

## Return Value

If a function was found and called it returns 1 (true), otherwise 0 (false).

## Remarks

Note that Function will not run if [#MaxThreads](#) limit is reached.

## Related

[ahkFunction](#), [ahkFindFunc](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
```

### AutoHotkey module.

```
DllCall(dllpath
"\ahktextdll", "Str", "#Persistent`nMyFunc(param)
{`nSleep 1000`nMsgBox `%
param`n}", "Str", "", "CDecl") ; start a new thread,
```

### just the function.

```
DllCall(dllpath
"\ahkPostFunction", "Str", "MyFunc", "Str", "Hello
World!", "Str", "", "Str", "", "Str", "", "Str", "", "Str",
"", "Str", "", "Str", "", "Str", "", "Str", "", "CDecl") ;
```

### call the function.

```
Sleep 2000 ; wait 2 seconds and exit
```

; Same example like above using included

### AutoHotkey.dll

```
dll:=AhkThread("#Persistent`nMyFunc(param)
{`nSleep 1000`nMsgBox `% param`n}")
dll.ahkPostFunction["MyFunc", "Hello World!"]
Sleep 2000
```

# ahkExec

Exported function that executes some code temporarily from string in currently executed script.

```
OutputVar := ahkExec(Code)
```

```
Command Example: ahkExec "MsgBox `% A_Now"
 DllCall
 "AutoHotkey.dll\ahkExec", "Str", "MsgBox `%
 A_Now", "Char"
 DllCall
 "AutoHotkey.exe\ahkExec", "Str", "MsgBox `%
 A_Now", "Char"
```

```
Function Example: Result := ahkExec("MsgBox `%
 A_Now")
 Result :=
 DllCall("AutoHotkey.dll\ahkExec", "Str",
 "MsgBox `% A_Now", "Char")
 Result :=
 DllCall("AutoHotkey.exe\ahkExec", "Str",
 "MsgBox `% A_Now", "Char")
```

## Parameters

### OutputVar

The name of the variable in which to store 1 (true) if code was created and executed successfully, 0 (false) otherwise.

## NewCode

Code to execute temporarily in currently running script.

## Return Value

If code was created and executed successfully 1 (true) is returned, otherwise 0 (false).

## Related

[ahkFindFunc](#), [ahkFindLabel](#), [ahkassign](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll","Str","", "Str","", "CDecl") ; start a
new thread, just the function.
DllCall(dllpath "\ahkExec","Str","MsgBox Hello
World!","CDecl") ; add and execute code

; Same example like above but using included
AutoHotkey.dll
dll:=AhkThread()
dll.ahkExec["MsgBox Hello World!"]

; Execute code in current thread
ahkExec("MsgBox Test")
```

# addScript

Exported function that adds new code from string to currently executed script.

```
OutputVar := addScript(NewCode [, WaitExecute])
```

```
Command Example: addScript "MsgBox `% A_Now",
1
DllCall
"AutoHotkey.dll\addScript", "Str", "MsgBox `%
A_Now", "Int", 1, "UPTR"
DllCall
"AutoHotkey.exe\addScript", "Str", "MsgBox `%
A_Now", "Int", 1, "UPTR"
Function Example: LinePtr := addScript("MsgBox
`% A_Now", 1)
LinePtr :=
DllCall("AutoHotkey.dll\addScript", "Str",
"MsgBox `% A_Now", "Int", 1, "UPTR")
LinePtr :=
DllCall("AutoHotkey.exe\addScript", "Str",
"MsgBox `% A_Now", "Int", 1, "UPTR")
```

## Parameters

### OutputVar

The name of the variable in which to store the line pointer to new code, if code could not be added 0 will be stored.

### NewCode

New Code to add to currently running script.

### WaitExecute (optional)

0 = add code but do not execute.

1 = add code, execute and wait for return.

2 = add code, execute and return immediately (don't wait for code to end execution).

## Related

[ahkFindFunc](#), [addFile](#), [ahkFindLabel](#), [ahkassign](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "", "Str", "", "Cdecl") ; start
new, empty thread.
DllCall(dllpath "\addScript", "Str", "MsgBox Hello
World!", "Int", 1, "Cdecl") ; add and execute code

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread()
dll.addScript["MsgBox Hello World!", 1]

; Add and execute script in current thread
addScript("MsgBox Test", 1)
```

# addFile

Adds new code from a file on disk or network in currently executed script.

```
OutputVar := addFile(FilePath [, WaitExecute])
```

```
Command Example: addFile A_ScriptDir
"\MyScript.ahk", 1
DllCall
"AutoHotkey.dll\addFile", "Str", A_ScriptDir
"\MyScript.ahk", "Int", 1, "UPTR"
DllCall
"AutoHotkey.exe\addFile", "Str", A_ScriptDir
"\MyScript.ahk", "Int", 1, "UPTR"
```

```
Function Example: LinePtr :=
addFile(A_ScriptDir "\MyScript.ahk", 1)
OutputVar :=
DllCall("AutoHotkey.dll\addFile", "Str",
A_ScriptDir "\MyScript.ahk", "Int", 1, "UPTR")
OutputVar :=
DllCall("AutoHotkey.exe\addFile", "Str",
A_ScriptDir "\MyScript.ahk", "Int", 1, "UPTR")
```

## Parameters

### OutputVar

The name of the variable in which to store the line pointer to new code, if code could not be added 0 will be stored.

### NewCode

New AutoHotkey script saved on disk or network to be added to currently running script.

### WaitExecute (optional)

0 = add code but do not execute it.

1 = add code, execute it and wait for it to finish execution.

2 = add code, execute it and return immediately (does not wait until execution finished).

## Related

[ahkFindFunc](#), [addScript](#), [ahkFindLabel](#), [ahkassign](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "", "Str", "", "Cdecl") ; start
new, empty thread.
DllCall(dllpath
"\addFile", "Str", "NewScript.ahk", "Int", 1, "Cdecl")
; add and execute code

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread()
dll.addFile["NewScript.ahk", 1]

; Add new script to current thread
addFile("NewScript.ahk", 1)
```



# ahkPause

Exported function used to pause or un-pause AutoHotkey module.

Parameters can be strings only.

```
OutputVar := DllCall("Module\ahkPause", "Str",
"On|Off", "Int")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkPause", "PTR", 1, "Int"
DllCall
"AutoHotkey.exe\ahkPause", "PTR", 0, "Int"
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkPause", "Str", "On",
"Int")
OutputVar :=
DllCall("AutoHotkey.exe\ahkPause", "Str",
"Off", "Int")
```

## Parameters

### OutputVar

The name of the variable to store 1 / true if script is paused or 0 / false otherwise.

On / Off / TRUE / FALSE / 1 / 0

Parameter can be string for On / Off or PTR for TRUE / FALSE / NULL /

1 / 0 where 1 (true) means On.

## Related

[ahkPostFunction](#), [ahkFindFunc](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath "\ahktextdll","Str",
(
#Persistent
Loop
 ToolTip `% A_TickCount
), "Str", "", "CDecl") ; start a new thread, just
the function.
Sleep 1000
DllCall(dllpath "\ahkPause","Str","On","CDecl") ;
Pause thread.
MsgBox End

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread("
(
#Persistent
Loop
 ToolTip `% A_TickCount
)")
Sleep 1000
dll.ahkPause["On"]
MsgBox End
```

# ahkReady

Exported function to check if a script is currently executed by AutoHotkey module.

**This function is only available in AutoHotkey.dll**

```
OutputVar := DllCall("AutoHotkey.dll\ahkReady")
```

```
Function Example: IsPaused :=
DllCall("AutoHotkey.dll\ahkReady")
```

## OutputVar

The name of the variable to store 1 / true if script is executed or 0 / false otherwise.

## Related

[ahkPostFunction](#), [ahkFindFunc](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath "\ahktextdll", "Str", "Sleep
2000", "Str", "", "CDecl") ; start a new thread, just
the function.
```

```
While DllCall(dllpath "\ahkReady") ; wait until
AutoHotkey.dll exits.
 Sleep 100
MsgBox End
```

```
; Same example like above using included
Autohotkey.dll
dll:=AhkThread("Sleep 2000")
while dll.ahkReady()
 Sleep 100
MsgBox End
```

# ahkReload

Exported function to reload currently executed script by AutoHotkey module.

**This function is only available in AutoHotkey.dll**

```
OutputVar := DllCall("AutoHotkey.dll\ahkReload",
"Int", Timeout, "Int")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkReload", "Int", 0, "Int"
```

```
Function Example:
DllCall("AutoHotkey.dll\ahkReload", "Int", 0,
"Int")
```

## Parameters

### OutputVar

The name of the variable to store always 0 / false so can be ignored.

### Timeout

Time to wait until thread exits, use FALSE / NULL / 0 = no Timeout.

NOTE: using a **negative** value you can force exit after given Timeout, with a **positive** Timeout it might take longer to exit thread.

## Related

ahkPostFunction, ahkFindFunc, DllCall

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "#Persistent`nMyFunc(param)
{`nSleep 10000`nMsgBox `%
param`n}", "Str", "", "CDecl") ; start a new thread,
just the function.
DllCall(dllpath
"\ahkFunction", "Str", "MyFunc", "Str", "Hello
World!", "CDecl Str") ; call the function.
```

# ahkTerminate

Exported function that exits script currently executed by AutoHotkey module.

**This function is only available in AutoHotkey.dll**

```
OutputVar := DllCall("AutoHotkey.dll\ahkTerminate",
"Int", Timeout, "Int")
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkTerminate", "Int", 500,
"Int"
Function Example:
DllCall("AutoHotkey.dll\ahkTerminate", "Int",
500, "Int")
```

## Parameters

### OutputVar

The name of the variable to store always 0 / false so can be ignored.

### Timeout

Time to wait until thread exits, use FALSE / NULL / 0 = no Timeout.  
NOTE: using a **negative** value you can force exit after given Timeout,  
with a **positive** Timeout it might take longer to exit thread but assures a  
clean exit.

## Related

[ahkPostFunction](#), [ahkFindFunc](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary", "Str", dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll", "Str", "MsgBox", "Str", "", "CDecl") ;
start a new thread.
Sleep 1000
DllCall(dllpath "\ahkTerminate", "UInt", 0, "CDecl")
; terminate thread.
MsgBox End

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread("MsgBox")
Sleep 1000
dll.ahkterminate[]
MsgBox End
```

# ahkFindFunc

Exported function that finds a function in currently executed script and returns a pointer to it.

```
OutputVar := ahkFindFunc("FuncName")
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkFindFunc", "Str",
"FuncName", "PTR")
OutputVar :=
DllCall("AutoHotkey.exe\ahkFindFunc", "Str",
"FuncName", "PTR")
```

## Parameters

### OutputVar

The name of the variable to store the function pointer in or 0 if the function was not found.

### FuncName

The name of the function to find.

## Related

[ahkFunction](#), [ahkPostFunction](#), [ahkFindLabel](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll","Str","#Persistent`nMyFunc(param)
{`nReturn param`n}","Str","", "CDecl") ; start a
new thread, just the function.
MsgBox % DllCall(dllpath
"\ahkFindFunc","Str","MyFunc","CDecl PTR") ; call
the function and display pointer in MsgBox.

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread("#Persistent`nMyFunc(param)
{`nReturn param`n}")
MsgBox % dll.ahkFindFunc["MyFunc"]

; Find pointer to function in current thread
MsgBox % ahkFindFunc("NumGet")
```

# ahkFindLabel

Exported function that finds a label in currently executed script and returns a pointer to it.

```
OutputVar := ahkFindLabel("LabelName")
```

```
Function Example: OutputVar :=
DllCall("AutoHotkey.dll\ahkFindLabel", "Str",
"LabelName", "PTR")
OutputVar :=
DllCall("AutoHotkey.exe\ahkFindLabel", "Str",
"LabelName", "PTR")
```

## Parameters

### OutputVar

The name of the variable to store the label pointer in or 0 if the label was not found.

### LabelName

Name of the label to find.

## Related

[ahkFunction](#), [ahkPostFunction](#), [ahkFindLabel](#), [DllCall](#)

## Examples

```
dllpath:=A_AhkDir "\AutoHotkey.dll"
DllCall("LoadLibrary","Str",dllpath) ; Load the
AutoHotkey module.
DllCall(dllpath
"\ahktextdll","Str","#Persistent`nMyLabel:`nReturn
","Str","", "CDecl") ; start a new thread, just the
function.
MsgBox % DllCall(dllpath
"\ahkFindLabel","Str","MyLabel","CDecl PTR") ;
call the function and display pointer in MsgBox.

; Same example like above using included
AutoHotkey.dll
dll:=AhkThread("#Persistent`nMyLabel:`nReturn")
MsgBox % dll.ahkFindLabel["MyLabel"]

; Find label in current thread
MsgBox % ahkFindLabel("MyLabel")
```

# ahkExecuteLine

Exported function that executes script from given line pointer.

```
OutputVar := ahkExecuteLine(LinePointer, Mode, Wait)
```

```
Command Example: DllCall
"AutoHotkey.dll\ahkExecuteLine", "PTR",
LinePointer, "UInt", 0, "UInt", 0, "PTR"
DllCall
"AutoHotkey.exe\ahkExecuteLine", "PTR",
LinePointer, "UInt", 0, "UInt", 0, "PTR"
```

```
Function Example: LinePointer :=
DllCall("AutoHotkey.dll\ahkExecuteLine", "PTR",
LinePointer, "UInt", 0, "UInt", 0, "PTR")
VarPointer :=
DllCall("AutoHotkey.exe\ahkExecuteLine", "PTR",
LinePointer, "UInt", 0, "UInt", 0, "PTR")
```

## Parameters

### OutputVar

The name of variable in which to store the pointer to next line.

### LinePointer

Pointer to Line to start execution from. NULL / false / 0 can be used to get pointer to first line.

### Mode

0 - will not execute and return a pointer to next line only.

1 - UNTIL\_RETURN

2 - UNTIL\_BLOCK\_END

3 - ONLY\_ONE\_LINE

## Wait

1 (true) if you want to wait until execution finishes, else FALSE / NULL / 0.

## Related

[ahkFindFunc](#), [ahkFindLabel](#), [ahkassign](#), [DllCall](#)

## Examples

```
MsgBox first Line %A_TickCount%
dll:=AhkThread("#Persistent`nMyVar:=`"Hello
World!`" `nReturn`nMsgBox `% MyVar") ; start a new
thread and set MyVar variable.
hLine:=0
Loop 3 ; Retrieve pointer to 4-th executable line.
 hLine := dll.ahkExecuteLine[hLine]
dll.ahkExecuteLine[hLine,1,1] ; execute from the
4-th line

; ----- Advanced Exaple -----
-

; AHK Structures
global _AHKDerefType := "LPTSTR marker, {_AHKVar
```



```

{struct{_AHKLine *mJumpToLine,_AHKFuncParam
*mParam,PTR *mClass},struct{PTR mBIF,UCHAR
*mOutputVars,PTR mID}},Int
mParamCount,MinParams,_AHKLabel
*mFirstLabel,*mLastLabel,_AHKVar
mGlobalvar,mVar,**mLazyVar,**mStaticVar,**mSta
ticLazyVar,Int
mGlobalVarCount,mVarCount,mVarCountMax,mLazyVarCou
nt,mStaticVarCount,mStaticLazyVarCount,Instances,U
CHAR mDefaultVarType,bool
mIsBuiltIn,mIsVariadic,mHasReturn"
global _AHKVar := "{Int64 mContentsInt64,Double
mContentsDouble,PTR mobject,PTR mVV},{char
*mByteContents,LPTSTR mCharContents},{UINT_PTR
mLength,_AHKVar *mAliasFor},{UINT_PTR
mCapacity,UINT_PTR mBIV},BYTE
mHowAllocated,mAttrib,mScope,mType,LPTSTR mName"

```

```

hLine:=ahkExecuteLine() ; get pointer to first
line in current script

```

```

mLine:=Struct(_AHKLine,hLine)

```

```

; Check if this is our script and correct line
number as it might be included file or static line
that is executes before script starts

```

```

While mLine.mFileIndex!=0 || mLine.mLineNumber!=1
 mLine[] := hLine := ahkExecuteLine(hLine) ;

```

```

reassign next line and update Structure

```

```

ahkExecuteLine(hLine,3,1) ; execute first line in
current thread

```

# ClipWait

Waits until the [clipboard](#) contains data.

```
ClipWait [SecondsToWait, 1]
```

```
Command Example: ClipWait, 0.2
Function Example: ClipWait(0.2)
```

## Parameters

### SecondsToWait

If omitted, the command will wait indefinitely. Otherwise, it will wait no longer than this many seconds (can contain a decimal point). Specifying 0 is the same as specifying 0.5.

### 1

If this parameter is omitted, the command is more selective, waiting specifically for text or files to appear ("text" includes anything that would produce text when you paste into Notepad). If this parameter is 1, the command waits for data of any kind to appear on the clipboard.

## ErrorLevel

If the wait period expires, [ErrorLevel](#) will be set to 1. Otherwise (i.e. the clipboard contains data), ErrorLevel is set to 0.

## Remarks

It's better to use this command than a loop of your own that checks to see if this clipboard is blank. This is because the clipboard is never opened by this command, and thus it performs better and avoids any chance of interfering with another application that may be using the clipboard.

This command considers anything convertible to text (e.g. HTML) to be text. It also considers files, such as those copied in an Explorer window via Control-C, to be text. Such files are automatically converted to their filenames (with full path) whenever the clipboard variable (%clipboard%) is referred to in the script. See [Clipboard](#) for details.

When 1 is present as the last parameter, the command will be satisfied when any data appears on the clipboard. This can be used in conjunction with [ClipboardAll](#) to save non-textual items such as pictures.

While the command is in a waiting state, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

## Related

[Clipboard](#), [WinWait](#), [KeyWait](#)

## Example

```
clipboard := "" ; Empty the clipboard
Send, ^c
```



# EnvGet

Retrieves an environment variable.

```
OutputVar := EnvGet("EnvVarName")
```

```
Function Example: tmp := EnvGet("Temp")
```

## Parameters

### OutputVar

The name of the variable in which to store the string.

### EnvVarName

The name of the [environment variable](#) to retrieve. For example:

```
EnvGet, OutputVar, Path.
```

## Remarks

If the specified environment variable is empty or does not exist, *OutputVar* is made blank.

The operating system limits each environment variable to 32 KB of text.

This command exists because [normal script variables](#) are not stored in the environment. This is because performance would be worse and also because the OS limits environment variables to 32 KB.

## Related

[EnvSet](#), [environment variables](#), [EnvUpdate](#), [Run](#), [RunWait](#)

## Example

[EnvGet](#), [OutputVar](#), [LogonServer](#)

# EnvSet

Writes a value to a [variable](#) contained in the environment.

**EnvSet** EnvVar, Value

```
Command Example: EnvSet "Mytemp", A_Temp
"\MyTemp"
Function Example: EnvSet("Temp",A_Temp
"\MyTemp")
```

## Parameters

### EnvVar

Name of the [environment variable](#) to use, e.g. "COMSPEC" or "PATH".

### Value

Value to set the [environment variable](#) to.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

## Remarks

The operating system limits each environment variable to 32 KB of text.

An environment variable created or changed with this command will be

accessible only to programs the script launches via [Run](#) or [RunWait](#). See [environment variables](#) for more details.

This command exists because [normal script variables](#) are not stored in the environment. This is because performance would be worse and also because the OS limits environment variables to 32 KB.

## Related

[EnvGet](#), [environment variables](#), [EnvUpdate](#), [Run](#), [RunWait](#)

## Example

```
EnvSet, AutGUI, Some text to put in the variable.
```

# OnClipboardChange

Registers a [function](#) or [function object](#) to run whenever the clipboard's content changes.

**OnClipboardChange** Func [, **AddRemove**]

## Parameters

### Func

A function name or [function object](#) to call. The function's parameter and return value are described [below](#).

### AddRemove

One of the following values:

- 1** (the default): Call the function after any previously registered functions.
- 1**: Call the function before any previously registered functions.
- 0**: Do not call the function.

If an OnClipboardChange label exists, it is always called first.

## Func

*FunctionName*(Type)

### Type

Contains one of the following values:

**0** if the clipboard is now empty;

**1** if it contains something that can be expressed as text (this includes [files copied](#) from an Explorer window);

**2** if it contains something entirely non-text such as a picture.

### *Return Value*

If this is the last or only OnClipboardChange function, the return value is ignored. Otherwise, the function can return a non-zero integer to prevent subsequent functions from being called.

## Example

The following example is a working script. Whenever it is running, it will briefly display a ToolTip for each clipboard change.

```
OnClipboardChange("ClipChanged")
return

ClipChanged(Type) {
 ToolTip Clipboard data type: %Type%
 Sleep 1000
 ToolTip ; Turn off the tip.
}
```

## Remarks

If the clipboard changes while an OnClipboardChange function is already running, that notification event is lost. If this is undesirable, specify [Critical](#) as

the label's first line. However, this will also buffer/defer other [threads](#) (such as the press of a hotkey) that occur while the OnClipboardChange thread is running.

If the script itself changes the clipboard, its OnClipboardChange functions are typically not executed immediately; that is, commands immediately below the command that changed the clipboard are likely to execute beforehand. To force the functions to execute immediately, use a short delay such as `Sleep 20` after changing the clipboard.

## Related

[Clipboard](#), [OnExit](#), [OnMessage](#), [RegisterCallback](#)

# SysGet

Retrieves system properties.

```
OutputVar := SysGet(Sub-command)
```

```
Function Example: VirtualScreenWidth :=
SysGet(78)
```

## Parameters

### OutputVar

The name of the variable in which to store the result.

### Sub-command

Specify one of the numbers from the table below to retrieve the corresponding value.

## Commonly Used

| Number | Description                                                                                                                               |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 80     | SM_CMONITORS: Number of display monitors on the desktop (not including "non-display pseudo-monitors").                                    |
| 43     | SM_CMOUSEBUTTONS: Number of buttons on mouse (0 if no mouse is installed).                                                                |
| 16, 17 | SM_CXFULLSCREEN, SM_CYFULLSCREEN: Width and height of the client area for a full-screen window on the primary display monitor, in pixels. |

|        |                                                                                                                                                                                                                                                                                     |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 61, 62 | SM_CXMAXIMIZED, SM_CYMAXIMIZED: Default dimensions, in pixels, of a maximized top-level window on the primary display monitor.                                                                                                                                                      |
| 59, 60 | SM_CXMAXTRACK, SM_CYMAXTRACK: Default maximum dimensions of a window that has a caption and sizing borders, in pixels. This metric refers to the entire desktop. The user cannot drag the window frame to a size larger than these dimensions.                                      |
| 28, 29 | SM_CXMIN, SM_CYMIN: Minimum width and height of a window, in pixels.                                                                                                                                                                                                                |
| 57, 58 | SM_CXMINIMIZED, SM_CYMINIMIZED: Dimensions of a minimized window, in pixels.                                                                                                                                                                                                        |
| 34, 35 | SM_CXMINTRACK, SM_CYMINTRACK: Minimum tracking width and height of a window, in pixels. The user cannot drag the window frame to a size smaller than these dimensions. A window can override these values by processing the WM_GETMINMAXINFO message.                               |
| 0, 1   | SM_CXSCREEN, SM_CYSCREEN: Width and height of the screen of the primary display monitor, in pixels. These are the same as the built-in variables <a href="#">A_ScreenWidth</a> and <a href="#">A_ScreenHeight</a> .                                                                 |
| 78, 79 | SM_CXVIRTUALSCREEN, SM_CYVIRTUALSCREEN: Width and height of the virtual screen, in pixels. The virtual screen is the bounding rectangle of all display monitors. The SM_XVIRTUALSCREEN, SM_YVIRTUALSCREEN metrics are the coordinates of the top-left corner of the virtual screen. |
| 19     | SM_MOUSEPRESENT: Nonzero if a mouse is installed; zero otherwise.                                                                                                                                                                                                                   |
| 75     | SM_MOUSEWHEELPRESENT: Nonzero if a mouse with a wheel is installed; zero otherwise.                                                                                                                                                                                                 |
| 63     | SM_NETWORK: Least significant bit is set if a network is present; otherwise, it is cleared. The other bits are reserved for future use.                                                                                                                                             |
|        | SM_REMOTECONTROL: This system metric is used in a                                                                                                                                                                                                                                   |

|        |                                                                                                                                                                                                                                                                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8193   | Terminal Services environment. Its value is nonzero if the current session is remotely controlled; zero otherwise.                                                                                                                                                                                                                                              |
| 4096   | SM_REMOTESSION: This system metric is used in a Terminal Services environment. If the calling process is associated with a Terminal Services client session, the return value is nonzero. If the calling process is associated with the Terminal Server console session, the return value is zero. The console session is not necessarily the physical console. |
| 70     | SM_SHOWSOUNDS: Nonzero if the user requires an application to present information visually in situations where it would otherwise present the information only in audible form; zero otherwise.                                                                                                                                                                 |
| 8192   | SM_SHUTTINGDOWN: Nonzero if the current session is shutting down; zero otherwise. <b>Windows 2000:</b> The retrieved value is always 0.                                                                                                                                                                                                                         |
| 23     | SM_SWAPBUTTON: Nonzero if the meanings of the left and right mouse buttons are swapped; zero otherwise.                                                                                                                                                                                                                                                         |
| 76, 77 | SM_XVIRTUALSCREEN, SM_YVIRTUALSCREEN: Coordinates for the left side and the top of the virtual screen. The virtual screen is the bounding rectangle of all display monitors. By contrast, the SM_CXVIRTUALSCREEN, SM_CYVIRTUALSCREEN metrics (further above) are the width and height of the virtual screen.                                                    |

## Not Commonly Used

| Number | Description                                                                                                               |
|--------|---------------------------------------------------------------------------------------------------------------------------|
| 56     | SM_ARRANGE: Flags specifying how the system arranged minimized windows. See MSDN for more information.                    |
| 67     | SM_CLEANBOOT: Specifies how the system was started:<br>0 Normal boot<br>1 Fail-safe boot<br>2 Fail-safe with network boot |

|        |                                                                                                                                                                                                                                                                                                                     |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5, 6   | SM_CXBORDER, SM_CYBORDER: Width and height of a window border, in pixels. This is equivalent to the SM_CXEDGE value for windows with the 3-D look.                                                                                                                                                                  |
| 13, 14 | SM_CXCURSOR, SM_CYCURSOR: Width and height of a cursor, in pixels. The system cannot create cursors of other sizes.                                                                                                                                                                                                 |
| 36, 37 | SM_CXDOUBLECLK, SM_CYDOUBLECLK: Width and height of the rectangle around the location of a first click in a double-click sequence, in pixels. The second click must occur within this rectangle for the system to consider the two clicks a double-click. (The two clicks must also occur within a specified time.) |
| 68, 69 | SM_CXDRAG, SM_CYDRAG: Width and height of a rectangle centered on a drag point to allow for limited movement of the mouse pointer before a drag operation begins. These values are in pixels. It allows the user to click and release the mouse button easily without unintentionally starting a drag operation.    |
| 45, 46 | SM_CXEDGE, SM_CYEDGE: Dimensions of a 3-D border, in pixels. These are the 3-D counterparts of SM_CXBORDER and SM_CYBORDER.                                                                                                                                                                                         |
| 7, 8   | SM_CXFIXEDFRAME, SM_CYFIXEDFRAME (synonymous with SM_CXDLGFRAME, SM_CYDLGFRAME): Thickness of the frame around the perimeter of a window that has a caption but is not sizable, in pixels. SM_CXFIXEDFRAME is the height of the horizontal border and SM_CYFIXEDFRAME is the width of the vertical border.          |
| 83, 84 | SM_CXFOCUSBORDER, SM_CYFOCUSBORDER: Width (in pixels) of the left and right edges and the height of the top and bottom edges of a control's focus rectangle. <b>Windows 2000:</b> The retrieved value is always 0.                                                                                                  |
| 21, 3  | SM_CXHSCROLL, SM_CYHSCROLL: Width of the arrow bitmap on a horizontal scroll bar, in pixels; and height of a horizontal scroll bar, in pixels.                                                                                                                                                                      |
| 10     | SM_CXHTHUMB: Width of the thumb box in a horizontal scroll bar, in pixels.                                                                                                                                                                                                                                          |

|        |                                                                                                                                                                                                                                                                                           |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11, 12 | SM_CXICON, SM_CYICON: Default width and height of an icon, in pixels.                                                                                                                                                                                                                     |
| 38, 39 | SM_CXICONSPACING, SM_CYICONSPACING: Dimensions of a grid cell for items in large icon view, in pixels. Each item fits into a rectangle of this size when arranged. These values are always greater than or equal to SM_CXICON and SM_CYICON.                                              |
| 71, 72 | SM_CXMENUCHECK, SM_CYMENUCHECK: Dimensions of the default menu check-mark bitmap, in pixels.                                                                                                                                                                                              |
| 54, 55 | SM_CXMENUSIZE, SM_CYMENUSIZE: Dimensions of menu bar buttons, such as the child window close button used in the multiple document interface, in pixels.                                                                                                                                   |
| 47, 48 | SM_CXMINSPPACING SM_CYMINSPPACING: Dimensions of a grid cell for a minimized window, in pixels. Each minimized window fits into a rectangle this size when arranged. These values are always greater than or equal to SM_CXMINIMIZED and SM_CYMINIMIZED.                                  |
| 30, 31 | SM_CXSIZE, SM_CYSIZE: Width and height of a button in a window's caption or title bar, in pixels.                                                                                                                                                                                         |
| 32, 33 | SM_CXSIZEFRAME, SM_CYSIZEFRAME: Thickness of the sizing border around the perimeter of a window that can be resized, in pixels. SM_CXSIZEFRAME is the width of the horizontal border, and SM_CYSIZEFRAME is the height of the vertical border. Synonymous with SM_CXFRAME and SM_CYFRAME. |
| 49, 50 | SM_CXSMICON, SM_CYSMICON: Recommended dimensions of a small icon, in pixels. Small icons typically appear in window captions and in small icon view.                                                                                                                                      |
| 52, 53 | SM_CXSMSIZE SM_CYSMSIZE: Dimensions of small caption buttons, in pixels.                                                                                                                                                                                                                  |
| 2, 20  | SM_CXVSCROLL, SM_CYVSCROLL: Width of a vertical scroll bar, in pixels; and height of the arrow bitmap on a vertical scroll bar, in pixels.                                                                                                                                                |
|        |                                                                                                                                                                                                                                                                                           |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4  | SM_CYCAPTION: Height of a caption area, in pixels.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 18 | SM_CYKANJIWINDOW: For double byte character set versions of the system, this is the height of the Kanji window at the bottom of the screen, in pixels.                                                                                                                                                                                                                                                                                                                            |
| 15 | SM_CYMENU: Height of a single-line menu bar, in pixels.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 51 | SM_CYSMCAPTION: Height of a small caption, in pixels.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 9  | SM_CYVTHUMB: Height of the thumb box in a vertical scroll bar, in pixels.                                                                                                                                                                                                                                                                                                                                                                                                         |
| 42 | SM_DBCSENABLED: Nonzero if User32.dll supports DBCS; zero otherwise.                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22 | SM_DEBUG: Nonzero if the debug version of User.exe is installed; zero otherwise.                                                                                                                                                                                                                                                                                                                                                                                                  |
| 82 | <p>SM_IMMENABLED: Nonzero if Input Method Manager/Input Method Editor features are enabled; zero otherwise.</p> <p>SM_IMMENABLED indicates whether the system is ready to use a Unicode-based IME on a Unicode application. To ensure that a language-dependent IME works, check SM_DBCSENABLED and the system ANSI code page. Otherwise the ANSI-to-Unicode conversion may not be performed correctly, or some components like fonts or registry setting may not be present.</p> |
| 87 | SM_MEDIACENTER: Nonzero if the current operating system is the Windows XP, Media Center Edition, zero if not.                                                                                                                                                                                                                                                                                                                                                                     |
| 40 | SM_MENUDROPALIGNMENT: Nonzero if drop-down menus are right-aligned with the corresponding menu-bar item; zero if the menus are left-aligned.                                                                                                                                                                                                                                                                                                                                      |
| 74 | SM_MIDEASTENABLED: Nonzero if the system is enabled for Hebrew and Arabic languages, zero if not.                                                                                                                                                                                                                                                                                                                                                                                 |
| 41 | SM_PENWINDOWS: Nonzero if the Microsoft Windows for Pen computing extensions are installed; zero otherwise.                                                                                                                                                                                                                                                                                                                                                                       |
| 44 | SM_SECURE: Nonzero if security is present; zero otherwise.                                                                                                                                                                                                                                                                                                                                                                                                                        |

|    |                                                                                                                                                                                                                                                                                                                                                               |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 81 | SM_SAMEDISPLAYFORMAT: Nonzero if all the display monitors have the same color format, zero otherwise. Note that two displays can have the same bit depth, but different color formats. For example, the red, green, and blue pixels can be encoded with different numbers of bits, or those bits can be located in different places in a pixel's color value. |
| 86 | SM_TABLETPC: Nonzero if the current operating system is the Windows XP Tablet PC edition, zero if not.                                                                                                                                                                                                                                                        |

## Remarks

SysGet simply calls the [GetSystemMetrics](#) function, which may support more sub-commands than are documented here.

## Related

[DllCall](#), [WinGet](#), [MonitorGet](#)

## Examples

```
SysGet, MouseButtonCount, 43
SysGet, VirtualScreenWidth, 78
SysGet, VirtualScreenHeight, 79
```

# DllCall()

Calls a function inside a DLL, such as a standard Windows API function.

```
OutputVar := DllCall("[DllFile\]Function" [, "Type1",
Arg1, "Type2", Arg2, "Cdecl ReturnType"])
```

```
Command Example: DllCall LoadLibrary, Str,
%A_AhkPath%
```

```
Function Example: hLib :=
DllCall("LoadLibrary", "Str", A_AhkPath)
```

## Parameters

### OutputVar

The name of the variable in which to store the the actual value returned by the function. If the function is of a type that does not return a value, the result is an undefined integer. If the function cannot be called due to an [error](#), the return value is blank (an empty string) and ErrorLevel is set.

### [DllFile\]Function

The DLL or EXE file name followed by a backslash and the name of the function. For example: `"MyDLL\MyFunction"` (the file extension ".dll" is the default when omitted). If an absolute path isn't specified, *DllFile* is assumed to be in the system's PATH or *A\_WorkingDir*.

*DllFile* may be omitted when calling a function that resides in User32.dll,

Kernel32.dll, ComCtl32.dll, or Gdi32.dll. For example,

"User32\IsWindowVisible" produces the same result as "IsWindowVisible".

If no function can be found by the given name, a "W" (Unicode) suffix is automatically appended. For example, "MessageBox" is the same as "MessageBoxW".

Performance can be dramatically improved when making *repeated* calls to a DLL by [loading it beforehand](#).

This parameter may also consist solely of an integer, which is interpreted as the address of the function to call. Sources of such addresses include [COM](#) and [RegisterCallback](#).

### Type1, Arg1

Each of these pairs represents a single parameter to be passed to the function. The number of pairs is unlimited. For *Type*, see the [types table](#) below. For *Arg*, specify the value to be passed to the function.

### Cdecl ReturnType

The word *Cdecl* is normally omitted because most functions use the standard calling convention rather than the "C" calling convention (functions such as `wsprintf` that accept a varying number of arguments are one exception to this). Note that most object-oriented C++ functions use the *thiscall* convention, which is not supported.

If present, the word *Cdecl* should be listed before the return type (if any). Separate each word from the next with a space or tab. For example:

```
"Cdecl Str".
```

Since a separate "C" calling convention does not exist in 64-bit code, *Cdecl* may be specified but has no effect on 64-bit builds of AutoHotkey.

*ReturnType*: If the function returns a 32-bit signed integer (Int), BOOL, or nothing at all, *ReturnType* may be omitted. Otherwise, specify one of the argument types from the [types table](#) below. The [asterisk suffix](#) is also supported.

## Types of Arguments and Return Values

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Str | <p>A string such as "Blue" or MyVar. If the called function modifies the string and the argument is a naked variable, its contents will be updated. For example, the following call would convert the contents of <i>MyVar</i> to uppercase: <code>DllCall("CharUpper", "Str", MyVar)</code>.</p> <p>However, if the function is designed to store a string larger than a variable's current capacity, ensure that the variable is large enough before calling the function. This can be achieved by calling <code>VarSetCapacity(MyVar, 123)</code>, where 123 is the length that <i>MyVar</i> must be able to hold.</p> <p>A <i>Str</i> argument must not be an <a href="#">expression</a> that evaluates to a number (such as <code>i+1</code>). If it is, the function is not called and <code>ErrorLevel</code> is set to -2.</p> <p>The <a href="#">asterisk variable</a> "Str*" is supported but rarely used. It can be used with functions that expect something like "TCHAR **" or</p> |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>"LPTSTR *".</p> <p>Note: When passing a string to a function, be aware what <i>type of string</i> the function expects.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| WStr  | <p>Since AutoHotkey uses UTF-16 natively, WStr (wide character string) is equivalent to Str.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| AStr  | <p>AStr causes the input value to be automatically converted to ANSI. Since the temporary memory used for this conversion is only large enough for the converted input string, any value written to it by the function is discarded. To receive an ANSI string as an output parameter, follow this example:</p> <pre> VarSetCapacity(buf, length) ; Allocate temporary buffer. DllCall("Function", "ptr", &amp;buf) ; Pass buffer to function. str := StrGet(&amp;buf, "cp0") ; Retrieve ANSI string from buffer. </pre> <p>See <a href="#">Script Compatibility</a> for equivalent Win32 types and other details.</p> |
| Int64 | <p>A 64-bit integer, whose range is -9223372036854775808 (-0x8000000000000000) to 9223372036854775807 (0x7FFFFFFFFFFFFFFF).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Int   | <p>A 32-bit integer (the most common integer type), whose range is -2147483648 (-0x80000000) to 2147483647 (0x7FFFFFFF). An Int is sometimes called a "Long".</p> <p>An Int should also be used for each BOOL argument expected by a function (a BOOL value should be either 1 or 0).</p> <p>An <i>unsigned</i> Int (UInt) is also used quite frequently, such as for</p>                                                                                                                                                                                                                                              |

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | DWORD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Short  | A 16-bit integer, whose range is -32768 (-0x8000) to 32767 (0x7FFF). An <b>unsigned</b> Short (UShort) can be used with functions that expect a WORD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Char   | An 8-bit integer, whose range is -128 (-0x80) to 127 (0x7F). An <b>unsigned</b> character (UChar) can be used with functions that expect a BYTE.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Float  | A 32-bit floating point number, which provides 6 digits of precision.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Double | A 64-bit floating point number, which provides 15 digits of precision.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Ptr    | <p>A <b>pointer-sized</b> integer, equivalent to Int or Int64 depending on whether the exe running the script is 32-bit or 64-bit. <i>Ptr</i> should be used for pointers to arrays or structures (such as RECT* or LPPOINT) and almost all handles (such as HWND, HBRUSH or HBITMAP). If the parameter is a pointer to a single numeric value such as LPDWORD or int*, generally the * or P suffix should be used instead of "Ptr".</p> <p><i>Ptr</i> can also be used with the * or P suffix; it should be used with functions that output a pointer via LPVOID* or similar.</p> <p><i>UPtr</i> is also valid, but is only unsigned in 32-bit builds as AutoHotkey does not support unsigned 64-bit integers.</p> <p>If compatibility with older versions of AutoHotkey is required, use a variable type as shown below:</p> <pre> Ptr := A_PtrSize ? "Ptr" : "UInt" ; If A_PtrSize is not defined, use UInt instead. DllCall("DeleteFile", Ptr, &amp;filename) ; Omit the quote marks around Ptr. </pre> |

Note: To pass a **NULL** handle or pointer, pass the integer 0.

Append an asterisk (with optional preceding space) to any of the above types to cause the address of the argument to be passed rather than the value itself (the called function must be designed to accept it). Since the value of such an argument might be modified by the function, whenever a naked variable is passed as the argument, that variable's contents will be updated. For example, the following call would pass the contents of MyVar to MyFunction by address, but would also update MyVar to reflect any changes made to it by MyFunction: `DllCall("MyDll\MyFunction", "Int*", MyVar)`.

In general, an asterisk is used whenever a function has an argument type or return type that starts with "LP". The most common example is LPDWORD, which is a pointer to a DWORD. Since a DWORD is an unsigned 32-bit integer, use "UInt\*" or "UIntP" to represent LPDWORD. An asterisk should not be used for string types such as LPTSTR, pointers to structures such as LPRECT, or arrays; for these, "Str" or "Ptr" should be used, depending on whether you pass a variable or its address.

Note: "Char\*" is not the same as "Str" because "Char\*" passes the address of an 8-bit number, whereas "Str" passes the address of a series of characters, which may be either 16-bit (Unicode) or 8-bit (for "AStr"), depending on the version of AutoHotkey. Similarly, "UInt\*" passes the address of a 32-bit number, so should not be used if the function expects an array of values or a structure larger than 32 bits.

Since variables in AutoHotkey have no fixed type, the address passed to the function points to temporary memory rather than the variable itself. It is not necessary to call `VarSetCapacity` on the variable as `DllCall` will update it correctly after the function returns.

\* or P  
(suffix)

U  
(prefix)

Prepend the letter U to any of the integer types above to interpret it as an unsigned integer (UInt64, UInt, UShort, and UChar). Strictly speaking, this is necessary only for return values and [asterisk variables](#) because it does not matter whether an argument passed by value is unsigned or signed (except for Int64).

If a negative integer is specified for an unsigned argument, the integer wraps around into the unsigned domain. For example, when -1 is sent as a UInt, it would become 0xFFFFFFFF.

*Unsigned* 64-bit integers produced by a function are not supported. Therefore, to work with numbers greater or equal to 0x8000000000000000, omit the U prefix and interpret any negative values received from the function as large integers. For example, a function that yields -1 as an Int64 is really yielding 0xFFFFFFFFFFFFFFFF if it is designed to yield a UInt64.

**Note:** When specifying an argument type or return type that does not contain a space or asterisk, the quotes around it may be omitted. For example, `Str` can be used in place of `"Str"` and `CDecl` in place of `"CDecl"`. In addition, the letter P may be used in place of asterisk to allow the quotes to be omitted there as well. For example: `UIntP`.

## Errors

DllCall throws an [exception](#) under any of the following conditions:

- The `[DllFile\]Function` parameter is a floating point number. A string or positive integer is required.
- The [return type](#) or one of the specified [arg types](#) is invalid. This error can also be caused by passing an [expression](#) that evaluates to a number to a

string (`str`) argument.

- The specified *DllFile* could not be accessed or loaded. If no explicit path was specified for *DllFile*, the file must exist in the system's PATH or `A_WorkingDir`. This error might also occur if the user lacks permission to access the file, or if AutoHotkey is 32-bit and the DLL is 64-bit or vice versa.
- The specified function could not be found inside the DLL.
- The function was called but it aborted with a fatal exception.

`Exception.Extra` contains the exception code. For example, `0xC0000005` means "access violation". In such cases, the thread is aborted (if `try` is not used), but any `asterisk variables` are still updated. An example of a fatal exception is dereferencing an invalid pointer such as `NULL (0)`. Since a `Cdecl` function never produces the error described in the next paragraph, it may generate an exception when too few arguments are passed to it.

- The function was called but was passed too many or too few arguments.

`Exception.Extra` contains the number of bytes by which the argument list was incorrect. If it is positive, too many arguments (or arguments that were too large) were passed, or the call requires `CDecl`. If it is negative, too few arguments were passed. This situation should be corrected to ensure reliable operation of the function. The presence of this error may also indicate that an exception occurred. Note that due to the x64 calling convention, 64-bit builds never raise this error.

## Exceptions and `A_LastError`

In spite of the built-in exception handling, it is still possible to crash a script with `DllCall`. This can happen when a function does not directly generate an exception but yields something inappropriate, such as a bad pointer or a string that is not terminated. This might not be the function's fault if the script passed it an unsuitable value such as a bad pointer or a `str` with insufficient capacity. A script can also crash when it specifies an inappropriate argument type or return type, such as claiming that an ordinary integer yielded by a function is an `asterisk variable` or `str`.

The built-in variable `A_LastError` contains the result of the operating system's `GetLastError()` function, which is called immediately after the function is called (this has no measurable impact on performance). `A_LastError` is a number between 0 and 4294967295. Like `ErrorLevel`, `A_LastError` is a per-thread setting; that is, interruptions by other threads cannot change it. However, `A_LastError` is also set by `Run/RunWait` and some other commands.

## Performance

When making repeated calls to a DLL, performance can be dramatically improved by loading it explicitly (*this is not necessary for a `standard DLL` such as `User32` because it is always resident*). This practice avoids the need for `DllCall` to internally call `LoadLibrary` and `FreeLibrary` each time. For example:

```
hModule := DllCall("LoadLibrary", "Str",
"MyFunctions.dll", "Ptr") ; Avoids the need
for DllCall() in the loop to load the library.
Loop, Files, C:\My Documents*.*, R
 result := DllCall("MyFunctions\BackupFile",
```

```
"Str", A_LoopFilePath)
DllCall("FreeLibrary", "Ptr", hModule) ; To
conserve memory, the DLL may be unloaded after
using it.
```

Even faster performance can be achieved by looking up the function's address beforehand. For example:

```
; In the following example, if the DLL isn't
yet loaded, use LoadLibrary in place of
GetModuleHandle.
MulDivProc := DllCall("GetProcAddress", Ptr,
DllCall("GetModuleHandle", Str, "kernel32",
"Ptr"), AStr, "MulDiv", "Ptr")
Loop 500
 DllCall(MulDivProc, Int, 3, Int, 4, Int, 3)
```

If DllCall's first parameter is a literal string such as "MulDiv" and the DLL containing the function is ordinarily loaded before the script starts, the string is automatically resolved to a function address. This built-in optimization is more effective than the example shown above.

Finally, when passing a string-variable to a function that will not change the length of the string, performance is improved by passing the variable by address (e.g. &MyVar) rather than as a "str" (especially when the string is very long).

The following example converts a string to uppercase:

```
DllCall("CharUpper", Ptr, &MyVar, Ptr).
```

## Structures and Arrays

A structure is a collection of *members* (fields) stored adjacently in memory. Most members tend to be integers.

Functions that accept the address of a structure (or a memory-block array) can be called by storing the structure's raw binary data in a normal variable. The following steps are generally used:

1) Call `VarSetCapacity(MyStruct, 123, 0)` to ensure that the target variable is large enough to hold the structure's data. Replace 123 with a number that is at least as large as the size of the structure. Specifying zero as the last parameter is optional; it initializes all members to be binary zero, which is typically used to avoid calling `NumPut()` as often in the next step.

2) If the target function uses the values initially in the structure, call `NumPut(123, MyStruct, 4, "UInt")` to initialize any members that should be non-zero. Replace 123 with the integer to be put into the target member (or specify `&Var` to store the [address](#) of a variable). Replace 4 with the offset of the target member (see step #4 for description of "offset"). Replace "UInt" with the appropriate type or omit it if the member is a pointer or handle.

3) Call the target function, passing the [address](#) of `MyStruct` as a `UInt` or `Ptr` argument. For example, `DllCall("MyDll\MyFunc", Ptr, &MyStruct)`. The function will examine and/or change some of the members.

4) Use `MyInteger := NumGet(MyStruct, 4, "UInt")` to retrieve any desired integers from the structure. Replace 4 with the offset of the target member in the structure. The first member is always at offset 0. The second

member is at offset 0 plus the size of the first member (typically 4). Members beyond the second are at the offset of the previous member plus the size of the previous member. Most members -- such as DWORD, Int, and [other types of 32-bit integers](#) -- are 4 bytes in size. Replace "UInt" with the appropriate type or omit it if the member is a pointer or handle.

See [Structure Examples](#) for actual usages.

## Known Limitations

When a [variable's address](#) (e.g. `&MyVar`) is passed to a function and that function alters the length of the variable's contents, subsequent uses of the variable may behave incorrectly. To fix this, do one of the following: 1) Pass `MyVar` as a "Str" argument rather than as a Ptr/address; 2) Call [VarSetCapacity\(MyVar, -1\)](#) to update the variable's internally-stored length after calling `DllCall`.

Any binary zero stored in a variable by a function hides all data to the right of the zero; that is, such data cannot be accessed or changed by most commands and functions. However, such data can be manipulated by the [address operator](#) and [NumPut/NumGet](#), as well as `DllCall` itself.

A function that returns the address of one of the strings that was passed into it might return an identical string in a different memory address than expected. For example calling `CharLower(CharUpper(MyVar))` in a programming language would convert `MyVar`'s contents to lowercase. But when the same is done with `DllCall()`, `MyVar` would be uppercase after the following call because

CharLower would have operated on a different/temporary string whose contents were identical to *MyVar*:

```
MyVar := "ABC"
result := DllCall("CharLower", str
DllCall("CharUpper", Str, MyVar, Str), Str)
```

To work around this, change the two underlined "Str" values above to Ptr. This interprets CharUpper's return value as a pure address that will get passed as an integer to CharLower.

Certain limitations may be encountered when dealing with strings. For details, see [Script Compatibility](#).

## Component Object Model (COM)

COM objects which are accessible to VBScript and similar languages are typically also accessible to AutoHotkey via [ComObjCreate](#), [ComObjGet](#) or [ComObjActive](#) and the built-in [object syntax](#).

COM objects which don't support [IDispatch](#) can be used with DllCall by retrieving the address of a function from the virtual function table of the object's interface. For more details, see [the example](#) further below.

Much of the .NET Framework is also accessible via COM and DllCall. See <http://www.autohotkey.com/forum/topic26191.html>.

## Related

[Script Compatibility](#), [PostMessage](#), [OnMessage](#), [RegisterCallback](#), [Run](#), [VarSetCapacity](#), [Functions](#), [SysGet](#), [MSDN Library](#)

## Examples

**; Example: Calls the Windows API function "MessageBox" and report which button the user presses.**

```
WhichButton := DllCall("MessageBox", "Int", "0",
"Str", "Press Yes or No", "Str", "Title of box",
"Int", 4)
MsgBox You pressed button #%WhichButton%.
```

**; Example: Changes the desktop wallpaper to the specified bitmap (.bmp) file.**

```
DllCall("SystemParametersInfo", UInt, 0x14, UInt,
0, Str, A_WinDir . "\winnt.bmp", UInt, 2)
```

**; Example: Calls the API function "IsWindowVisible" to find out if a Notepad window is visible.**

```
DetectHiddenWindows On
if not DllCall("IsWindowVisible", "Ptr",
WinExist("Untitled - Notepad")) ; WinExist()
returns an HWND.
```

```
MsgBox The window is not visible.
```

**; Example: Calls the API's wsprintf() to pad the number 432 with leading zeros to make it 10 characters wide (0000000432).**

```
VarSetCapacity(ZeroPaddedNumber, 20) ; Ensure the
variable is large enough to accept the new string.
DllCall("wsprintf", "Str", ZeroPaddedNumber,
"Str", "%010d", "Int", 432, "Cdecl") ; Requires
the Cdecl calling convention.
MsgBox %ZeroPaddedNumber%
```

```
; Example: Demonstrates QueryPerformanceCounter(),
which gives more precision than A_TickCount's
10ms.
```

```
DllCall("QueryPerformanceCounter", "Int64*",
CounterBefore)
Sleep 1000
DllCall("QueryPerformanceCounter", "Int64*",
CounterAfter)
MsgBox % "Elapsed QPC time is " . CounterAfter -
CounterBefore
```

```
; Example: This is a hotkey that temporarily
reduces the mouse cursor's speed, which
facilitates precise positioning.
; Hold down the F1 key to slow down the cursor.
Release it to return to original speed.
```

```
F1::
SPI_GETMOUSESPEED := 0x70
SPI_SETMOUSESPEED := 0x71
; Retrieve the current speed so that it can be
restored later:
DllCall("SystemParametersInfo", UInt,
SPI_GETMOUSESPEED, UInt, 0, UIntP, OrigMouseSpeed,
UInt, 0)
; Now set the mouse to the slower speed specified
```

in the next-to-last parameter (the range is 1-20, 10 is default):

```
DllCall("SystemParametersInfo", UInt,
SPI_SETMOUSESPEED, UInt, 0, Ptr, 3, UInt, 0)
KeyWait F1 ; This prevents keyboard auto-repeat
from doing the DllCall repeatedly.
return
```

```
F1 up::DllCall("SystemParametersInfo", UInt, 0x71,
UInt, 0, Ptr, OrigMouseSpeed, UInt, 0) ; Restore
the original speed.
```

; Example: Monitors the active window and display  
the position of its vertical scroll bar in its  
; focused control (with real-time updates).

```
SetTimer, WatchScrollBar, 100
return
```

```
WatchScrollBar:
```

```
ActiveWindow := WinExist("A")
```

```
if not ActiveWindow ; No active window.
return
```

```
FocusedControl := ControlGetFocus() ; Omit all
parameter to use the Last Found Window.
```

```
if not FocusedControl ; No focused control.
return
```

; Display the vertical or horizontal scroll bar's  
position in a Tooltip:

```
ChildHWND := ControlGetHwnd(FocusedControl)
```

```
ToolTip(DllCall("GetScrollPos", "Ptr", ChildHWND,
"Int", 1)) ; Last parameter is 1 for SB_VERT, 0
for SB_HORZ.
```

```
return
```

; Example: This is a working script that writes some text to a file then reads it back into memory.

; This method can be used to help performance in cases where multiple files are being read or written simultaneously.

; Alternatively, FileOpen can be used to the same effect.

```
FileSelect, FileName, S16,, Create a new file:
```

```
if FileName = ""
```

```
 return
```

```
 GENERIC_WRITE := 0x40000000 ; Open the file for writing rather than reading.
```

```
 CREATE_ALWAYS := 2 ; Create new file (overwriting any existing file).
```

```
 hFile := DllCall("CreateFile", Str, FileName, UInt, GENERIC_WRITE, UInt, 0, Ptr, 0, UInt, CREATE_ALWAYS, UInt, 0, Ptr, 0, Ptr)
```

```
 if not hFile
```

```
 {
```

```
 MsgBox Can't open "%FileName%" for writing.
```

```
 return
```

```
 }
```

```
 TestString := "This is a test string.`r`n" ; When writing a file this way, use `r`n rather than `n to start a new line.
```

```
 DllCall("WriteFile", Ptr, hFile, Str, TestString, UInt, StrLen(TestString), UIntP, BytesActuallyWritten, Ptr, 0)
```

```
 DllCall("CloseHandle", Ptr, hFile) ; Close the file.
```

; Now that the file was written, read its contents back into memory.

```
 GENERIC_READ := 0x80000000 ; Open the file for
```

reading rather than writing.

OPEN\_EXISTING := 3 ; This mode indicates that the file to be opened must already exist.

FILE\_SHARE\_READ := 0x1 ; This and the next are whether other processes can open the file while we have it open.

FILE\_SHARE\_WRITE := 0x2

```
hFile := DllCall("CreateFile", Str, FileName,
 UInt, GENERIC_READ, UInt,
 FILE_SHARE_READ|FILE_SHARE_WRITE, Ptr, 0, UInt,
 OPEN_EXISTING, UInt, 0, Ptr, 0)
```

```
if not hFile
```

```
{
 MsgBox Can't open "%FileName%" for reading.
 return
}
```

; Make the variable empty for testing purposes, but ensure it retains sufficient capacity:

```
BytesToRead := VarSetCapacity(TestString,
 StrLen(TestString))
```

```
DllCall("ReadFile", Ptr, hFile, Str, TestString,
 UInt, BytesToRead, UIntP, BytesActuallyRead, Ptr,
 0)
```

```
DllCall("CloseHandle", Ptr, hFile) ; Close the file.
```

```
MsgBox The following string was read from the file: %TestString%
```

; Example: Hides the mouse cursor when you press Win+C. To later show the cursor, press Win+C again.

; This script is from

[www.autohotkey.com/forum/topic6107.html](http://www.autohotkey.com/forum/topic6107.html)

```
OnExit("ShowCursor") ; Ensure the cursor is made visible when the script exits.
```

```

return

ShowCursor()
{
 SystemCursor("On")
}

#c::SystemCursor("Toggle") ; Win+C hotkey to
toggle the cursor on and off.

SystemCursor(OnOff:=1) ; INIT = "I","Init"; OFF
= 0,"Off"; TOGGLE = -1,"T","Toggle"; ON = others
{
 static AndMask, XorMask, type, h_cursor
, c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13 ;
system cursors

b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, b13 ;
blank cursors

h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, h13 ;
handles of default cursors
 if (OnOff = "Init" or OnOff = "I" or type =
"") ; init when requested or at first call
 {
 type := "h"
; active default cursors
 VarSetCapacity(h_cursor, 4444, 1)
 VarSetCapacity(AndMask, 32*4, 0xFF)
 VarSetCapacity(XorMask, 32*4, 0)
 system_cursors :=
"32512,32513,32514,32515,32516,32642,32643,32644,3
2645,32646,32648,32649,32650"
 c := StrSplit(system_cursors, ",")
 Loop c.Length
 {

```

```

 h_cursor := DllCall("LoadCursor",
"Ptr",0, "Ptr",c[A_Index])
 h%A_Index% := DllCall("CopyImage",
"Ptr",h_cursor, "UInt",2, "Int",0, "Int",0,
"UInt",0)
 b%A_Index% := DllCall("CreateCursor",
"Ptr",0, "Int",0, "Int",0
, "Int",32, "Int",32,
"Ptr",&AndMask, "Ptr",&XorMask)
 }
}
if (OnOff = 0 or OnOff = "off" or type = "h"
and (OnOff < 0 or OnOff = "Toggle" or OnOff =
"T"))
 type := "b" ; use blank cursors
else
 type := "h" ; use the saved cursors

Loop c.Length
{
 h_cursor := DllCall("CopyImage",
"Ptr",%type%%A_Index%, "UInt",2, "Int",0, "Int",0,
"UInt",0)
 DllCall("SetSystemCursor",
"Ptr",h_cursor, "UInt",c[A_Index])
}
}

```

; Structure Example: Pass the address of a RECT structure to GetWindowRect(), which sets the structure's ; members to the positions of the left, top, right, and bottom sides of a window (relative to the screen).

Run Notepad

```
WinWait Untitled - Notepad ; This also sets the
"last found window" for use with WinExist() below.
VarSetCapacity(Rect, 16) ; A RECT is a struct
consisting of four 32-bit integers (i.e. 4*4=16).
DllCall("GetWindowRect", Ptr, WinExist(), Ptr,
&Rect) ; WinExist() returns an HWND.
MsgBox % "Left " . NumGet(Rect, 0, "Int") . " Top
" . NumGet(Rect, 4, "Int")
" . " Right " . NumGet(Rect, 8, "Int") . "
Bottom " . NumGet(Rect, 12, "Int")
```

; Structure Example: Pass to FillRect() the  
address of a RECT structure that indicates a part  
of the  
; screen to temporarily paint red.

```
VarSetCapacity(Rect, 16, 0) ; Set capacity to
hold four 4-byte integers and initialize them all
to zero.
NumPut(A_ScreenWidth//2, Rect, 8, "Int") ; The
third integer in the structure is "rect.right".
NumPut(A_ScreenHeight//2, Rect, 12, "Int") ; The
fourth integer in the structure is "rect.bottom".
hDC := DllCall("GetDC", "Ptr", 0, "Ptr") ; Pass
zero to get the desktop's device context.
hBrush := DllCall("CreateSolidBrush", "UInt",
0x0000FF, "Ptr") ; Create a red brush (0x0000FF
is in BGR format).
DllCall("FillRect", "Ptr", hDC, "Ptr", &Rect,
"Ptr", hBrush) ; Fill the specified rectangle
using the brush above.
DllCall("ReleaseDC", "Ptr", 0, "Ptr", hDC) ;
Clean-up.
DllCall("DeleteObject", "Ptr", hBrush) ; Clean-
up.
```

```
; Structure Example: Change the system's clock to
the specified date and time. Use caution when
; changing to a date in the future as it may cause
scheduled tasks to run prematurely!
```

```
SetSystemTime("20051008142211") ; Pass it a
timestamp (local, not UTC).
```

```
SetSystemTime(YYYYMMDDHHMISS)
; Sets the system clock to the specified date and
time.
```

```
; Caller must ensure that the incoming parameter
is a valid date-time stamp
; (local time, not UTC). Returns non-zero upon
success and zero otherwise.
```

```
{
; Convert the parameter from local time to UTC
for use with SetSystemTime().
```

```
UTC_Delta := DateDiff(A_Now, A_NowUTC,
"Seconds") ; Seconds is more accurate due to
rounding issue.
```

```
UTC_Delta := Round(-UTC_Delta/60) ; Round to
nearest minute to ensure accuracy.
```

```
YYYYMMDDHHMISS := DateAdd(YYYYMMDDHHMISS,
UTC_Delta, "Minutes") ; Apply offset to convert
to UTC.
```

```
VarSetCapacity(SystemTime, 16, 0) ; This
struct consists of 8 UShorts (i.e. 8*2=16).
```

```
Int := SubStr(YYYYMMDDHHMISS, 1, 4) ; YYYY
(year)
```

```
NumPut(Int, SystemTime, 0, "UShort")
Int := SubStr(YYYYMMDDHHMISS, 5, 2) ; MM
(month of year, 1-12)
```

```
NumPut(Int, SystemTime, 2, "UShort")
Int := SubStr(YYYYMMDDHHMISS, 7, 2) ; DD (day
```

```

of month)
 NumPut(Int, SystemTime, 6, "UShort")
 Int := SubStr(YYYYMMDDHHMISS, 9, 2) ; HH
(hour in 24-hour time)
 NumPut(Int, SystemTime, 8, "UShort")
 Int := SubStr(YYYYMMDDHHMISS, 11, 2) ; MI
(minute)
 NumPut(Int, SystemTime, 10, "UShort")
 Int := SubStr(YYYYMMDDHHMISS, 13, 2) ; SS
(second)
 NumPut(Int, SystemTime, 12, "UShort")

 return DllCall("SetSystemTime", Ptr,
&SystemTime)
}

```

```

/* More Structure Examples:

```

```

1) See the WinLIRC client script for a demonstration of how to use DllCall() to make a network connection to a TCP/IP server and receive data from it.

```

```

2) The operating system offers standard dialog boxes that prompt the user to pick a font and/or color, or an icon.

```

```

These dialogs use structures and are demonstrated at www.autohotkey.com/forum/topic17230.html.

```

```

*/

```

```

/*
Example: Temporarily remove the active window from the taskbar by using COM.

```

## Methods in ITaskbarList's VTable:

### IUnknown:

- 0 QueryInterface -- use ComObjQuery instead
- 1 AddRef -- use ObjAddRef instead
- 2 Release -- use ObjRelease instead

### ITaskbarList:

- 3 HrInit
- 4 AddTab
- 5 DeleteTab
- 6 ActivateTab
- 7 SetActiveAlt

```
*/
```

```
IID_ITaskbarList := "{56FDF342-FD6D-11d0-958A-006097C9A090}"
```

```
CLSID_TaskbarList := "{56FDF344-FD6D-11d0-958A-006097C9A090}"
```

```
; Create the TaskbarList object and store its
address in tbl.
```

```
tbl := ComObjCreate(CLSID_TaskbarList,
IID_ITaskbarList)
```

```
activeHwnd := WinExist("A")
```

```
DllCall(vtable(tbl,3), "ptr", tbl)
```

```
; tbl.HrInit()
```

```
DllCall(vtable(tbl,5), "ptr", tbl, "ptr",
activeHwnd) ; tbl.DeleteTab(activeHwnd)
```

```
Sleep 3000
```

```
DllCall(vtable(tbl,4), "ptr", tbl, "ptr",
activeHwnd) ; tbl.AddTab(activeHwnd)
```

```
; Non-dispatch objects must always be manually
freed.
```

```
ObjRelease(tbl)
```

```
vtable(ptr, n) {
```

```
 ; NumGet(ptr+0) returns the address of the
 object's virtual function
 ; table (vtable for short). The remainder of
 the expression retrieves
 ; the address of the nth function's address
 from the vtable.
 return NumGet(NumGet(ptr+0), n*A_PtrSize)
}
```

# DynaCall

Build in function, similar to DllCall but works with DllCall structures and uses Object syntax. It is often faster than DllCall, easier to use and it saves a lot of typing and code.

```
OutputVar := DynaCall("[DllFile\]Function",
"ParameterDefinition", Default1, Default2, Default3,
...)
```

```
Function Example: TrueSleep :=
DynaCall("kernel32\Sleep", "ui", 100)
Calling the Func: TrueSleep[1000], %TrueSleep%
(1000)
```

## Parameters

### OutputVar

The name of the variable in which to store the DynaCall object which is used to call the dll function.

### [DllFile\]Function

The DLL or EXE file name followed by a backslash and the name of the function. For example: "MyDLL\MyFunction" (".dll" is default and can be omitted). If an absolute path isn't specified, *DllFile* is assumed to be in the system's PATH or [A\\_WorkingDir](#).

*DllFile* may be omitted when calling a function that resides in User32.dll, Kernel32.dll, ComCtl32.dll, or Gdi32.dll. For example, "User32\IsWindowVisible" produces the same result as "IsWindowVisible".

If no function can be found by the given name, a "W" (Unicode) suffix is automatically appended. For example, "MessageBox" is the same as "MessageBoxW".

This parameter may also consist solely of an integer, which is interpreted as the address of the function to call. Sources of such addresses include [COM](#) and [RegisterCallback](#).

### ParameterDefinition

For definition you will need to use the short version of DllCall types.

| <b>DllCall</b>         | <b>DynaCall equivalent</b>                                      |
|------------------------|-----------------------------------------------------------------|
| Int                    | i                                                               |
| Str                    | s                                                               |
| AStr                   | a                                                               |
| WStr                   | w                                                               |
| Short                  | h                                                               |
| Char                   | c                                                               |
| Float                  | f                                                               |
| Double                 | d                                                               |
| PTR                    | t                                                               |
| Int64                  | i6                                                              |
| CDecl                  | Use == instead of =, for example "t==uis".                      |
| U prefix and *<br>or p | This is supported as well, for example: "ui=ui*s",<br>"ui=uips" |

The syntax for parameter definition is [ **Return type** [=] ]

**ParamType ParamType ParamType ...**

You can have any amount of space or tabs between parameters, those will be simply ignored.

- **Return type** and = or == are optional and can be left out.

### Changing order of parameters

It is possible to change the order of parameters to use when the function is called.

Therefore we need to use an object for parameter definition with following syntax.

[ **definition, FirstParam, SecondParam, ...**]

- Definition - Same as above
- FirstParam - First parameter to use when function is called
- SecondParam - Second parameter to use when function is called

This is best explained in following example.

```
AHKCMD:=DynaCall("SendMessage",
["t=tuitt", 3], A_ScriptHwnd, 0x111)
MsgBox % A_IsSuspended
AHKCMD[65305] ; suspend script,
following is the same call:
; DllCall("SendMessage", "PTR",
A_ScriptHwnd, "UInt", 0x111, "PTR",
65305, "PTR", 0)
MsgBox % A_IsSuspended
```

## Default Value

The default value to use when the parameter is omitted in function call. Note, this will be always the original order of parameters as specified in **ParameterDefinition**.

## Examples

```
AHKCMD:=DynaCall("SendMessage", ["t=tuitt", 3],
A_ScriptHwnd, 0x111) ; define a DynaCall
object.
AHKCMD[65305] ; call the actual function
AHKCMD.65305
%AHKCMD%(65305)
AHKCMD.PTR(65305) ; here (PTR) we can specify
any AHK return type.
```

## Related

[DllCall](#), [#DllImport](#), [WinApi](#)

# NumGet

Returns the binary number stored at the specified address+offset.

```
OutputVar := NumGet(VarOrAddress [, Offset = 0, Type = UPtr])
```

```
Function Example: val := NumGet(MyVar, "UInt")
```

## Parameters

### OutputVar

The name of the variable in which to store the number at the specified address+offset.

If the target address is invalid, an empty string is returned. However, some invalid addresses cannot be detected as such and may cause unpredictable behaviour.

### VarOrAddress

A memory address or variable. If *VarOrAddress* is a variable such as `MyVar` and it contains a string (not a pure number), the address of the variable's string buffer is used. This is usually equivalent to passing `&MyVar`, but omitting the "&" performs better and ensures that the target address + offset is [valid](#). If the variable contains a pure number, that number is assumed to be an address.

## Offset

An offset - in bytes - which is added to *VarOrAddress* to determine the target address.

## Type

One of the following strings (defaults to UPtr if omitted):  
UInt, Int, Int64, Short, UShort, Char, UChar, Double, Float, Ptr or UPtr

Unlike DllCall, these must be enclosed in quotes when used as literal strings.

For details see [DllCall Types](#).

## General Remarks

If only two parameters are present, the second parameter can be either *Offset* or *Type*. For example, `NumGet(var, "int")` is valid.

## Related

[NumPut](#), [DllCall](#), [VarSetCapacity](#)

# NumPut

Stores a number in binary format at the specified address+offset.

```
OutputVar := NumPut(Number, VarOrAddress [, Offset = 0, Type = UPtr])
```

```
Function Example: PTR := NumPut(100, var, "Uint")
```

## Parameters

### OutputVar

The name of the variable in which to store the the address to the right of the item just written is returned. This is often used when writing a sequence of numbers of different types, such as in a structure for use with DllCall..

If the target address is invalid, an empty string is returned. However, some invalid addresses cannot be detected as such and may cause unpredictable behaviour.

### Number

The number to store.

### VarOrAddress

A memory address or variable. If *VarOrAddress* is a variable such as

`MyVar` and it contains a string (not a pure number), the address of the variable's string buffer is used. This is usually equivalent to passing `&MyVar`, but omitting the "&" performs better and ensures that the target address + offset is *valid*. If the variable contains a pure number, that number is assumed to be an address.

### Offset

An offset - in bytes - which is added to *VarOrAddress* to determine the target address.

### Type

One of the following strings (defaults to UPtr if omitted):

UInt, Int, Int64, Short, UShort, Char, UChar, Double, Float, Ptr or UPtr

Unlike DllCall, these must be enclosed in quotes when used as literal strings.

For details see [DllCall Types](#).

## General Remarks

If an integer is too large to fit in the specified *Type*, its most significant bytes are ignored; e.g. `NumPut(257, var, 0, "Char")` would store the number 1.

If only three parameters are present, the third parameter can be either *Offset* or *Type*. For example, `NumPut(x, var, "int")` is valid.

## Related

NumGet, DllCall, VarSetCapacity

# RegisterCallback()

Creates a machine-code address that when called, redirects the call to a [function](#) in the script.

```
Address := RegisterCallback("FunctionName" , Options
:= "", ParamCount := FormalCount, EventInfo :=
Address)
```

## Parameters

### Address

Upon success, RegisterCallback() returns a numeric address that may be called by [DllCall](#) or anything else capable of calling a machine-code function. Upon failure, it returns an empty string. Failure occurs when *FunctionName*: 1) does not exist; 2) accepts too many or too few parameters according to *ParamCount*; or 3) accepts any [ByRef](#) parameters.

### FunctionName

A [function's](#) name, which must be enclosed in quotes if it is a literal string. This function is called automatically whenever *Address* is called. The function also receives the parameters that were passed to *Address*.

A [function reference](#) can be passed instead of a function name.

### Options

Specify zero or more of the following words. Separate each option from the next with a space (e.g. `C Fast`).

**Fast** or **F**: Avoids starting a new [thread](#) each time *FunctionName* is called. Although this performs better, it must be avoided whenever the thread from which *Address* is called varies (e.g. when the callback is triggered by an incoming message). This is because *FunctionName* will be able to change global settings such as [ErrorLevel](#), [A\\_LastError](#), and the [last-found window](#) for whichever thread happens to be running at the time it is called. For more information, see [Remarks](#).

**CDecl** or **C**: Makes *Address* conform to the "C" calling convention. This is typically omitted because the standard calling convention is much more common for callbacks.

### ParamCount

The number of parameters that *Address's* caller will pass to it. If entirely omitted, it defaults to the number of mandatory parameters in the [definition](#) of *FunctionName*. In either case, ensure that the caller passes exactly this number of parameters.

### EventInfo

An integer that *FunctionName* will see in [A\\_EventInfo](#) whenever it is called via this *Address*. This is useful when *FunctionName* is called by more than one *Address*. If omitted, it defaults to *Address*. Note: Unlike other global settings, the [current thread's](#) [A\\_EventInfo](#) is not disturbed by the [fast mode](#).

If the exe running the script is 32-bit, this parameter must be between 0 and 4294967295. If the exe is 64-bit, this parameter can be a 64-bit integer. Although `A_EventInfo` usually returns an unsigned integer, AutoHotkey does not fully support unsigned 64-bit integers and therefore some operations may cause the value to wrap into the signed range.

## The Callback Function's Parameters

A function assigned to a callback address may accept up to 31 parameters. Optional parameters are permitted, which is useful when the function is called by more than one caller.

Interpreting the parameters correctly requires some understanding of how the x86 calling conventions work. Since AutoHotkey does not have typed parameters, the callback's parameter list is assumed to consist of integers, and some reinterpretation may be required.

**AutoHotkey 32-bit:** All incoming parameters are unsigned 32-bit integers. Smaller types are padded out to 32 bits, while larger types are split into a series of 32-bit parameters.

If an incoming parameter is intended to be a signed integer, any negative numbers can be revealed by following either of the following examples:

```
; Method #1
if wParam > 0x7FFFFFFF
 wParam := -(-wParam) - 1

; Method #2: Relies on the fact that AutoHotkey
```

**natively uses signed 64-bit integers.**

```
wParam := wParam << 32 >> 32
```

**AutoHotkey 64-bit:** All incoming parameters are signed 64-bit integers. AutoHotkey does not natively support unsigned 64-bit integers. Smaller types are padded out to 64 bits, while larger types are always passed by address.

**AutoHotkey 32-bit/64-bit:** If an incoming parameter is intended to be 8-bit or 16-bit (or 32-bit on x64), the upper bits of the value might contain "garbage" which can be filtered out by using bitwise-and, as in the following examples:

```
Callback(UCharParam, UShortParam, UIntParam) {
 UCharParam &= 0xFF
 UShortParam &= 0xFFFF
 UIntParam &= 0xFFFFFFFF
 ; ...
}
```

If an incoming parameter is intended by its caller to be a string, what it actually receives is the address of the string. To retrieve the string itself, use [StrGet](#):

```
MyString := StrGet(MyParameter)
```

If an incoming parameter is the address of a structure, the individual members may be extracted by following the steps at [DllCall structures](#).

**Receiving parameters by address:** If the function is declared as [variadic](#), its final parameter is assigned the *address* of the first callback parameter which was not assigned to a script parameter. For example:

```

callback := RegisterCallback("TheFunc", "F", 3)
; Parameter list size must be specified.
TheFunc("TheFunc was called directly.")
; Call TheFunc directly.
DllCall(callback, float, 10.5, int64, 42)
; Call TheFunc via callback.
TheFunc(params*) {
 if IsObject(params)
 MsgBox % params[1]
 else
 MsgBox % NumGet(params+0, "float") ", "
 NumGet(params+A_PtrSize, "int64")
}

```

Most callbacks use the *stdcall* calling convention, which requires a fixed number of parameters. In those cases, *ParamCount* must be set to the size of the parameter list, where *Int64* and *Double* count as two 32-bit parameters.

With *Cdecl* or the 64-bit calling convention, *ParamCount* only affects how many script parameters are assigned values. If omitted, all optional parameters receive their default values and are excluded from the calculations for the address stored in *params*.

## What the Function Should Return

If the function uses [Return](#) without any parameters, or it specifies a blank value such as "" (or it never uses [Return](#) at all), 0 is returned to the caller of the callback. Otherwise, the function should return an integer, which is then returned to the caller. AutoHotkey 32-bit truncates return values to 32-bit, while AutoHotkey 64-bit supports 64-bit return values. Returning structs larger than

this (by value) is not supported.

## Fast vs. Slow

The default/slow mode causes the function to start off fresh with the default values for settings such as `SendMode` and `DetectHiddenWindows`. These defaults can be changed in the `auto-execute` section.

By contrast, the `fast mode` inherits global settings from whichever `thread` happens to be running at the time the function is called. Furthermore, any changes the function makes to global settings (including `ErrorLevel` and the `last-found window`) will go into effect for the `current thread`. Consequently, the fast mode should be used only when it is known exactly which thread(s) the function will be called from.

To avoid being interrupted by itself (or any other thread), a callback may use `Critical` as its first line. However, this is not completely effective when the function is called indirectly via the arrival of a message less than 0x312 (increasing `Critical's interval` may help). Furthermore, `Critical` does not prevent the function from doing something that might indirectly result in a call to itself, such as calling `SendMessage` or `DllCall`.

## Memory

Each use of `RegisterCallback()` allocates a small amount of memory (32 bytes plus system overhead). Since the OS frees this memory automatically when the script exits, any script that allocates a small, *fixed* number of callbacks does not

have to explicitly free the memory. By contrast, a script that calls RegisterCallback() an indefinite/unlimited number of times should explicitly call the following on any unused callbacks:

```
DllCall("GlobalFree", "Ptr", Address, "Ptr")
```

## Related

[DllCall](#), [OnMessage](#), [OnExit](#), [OnClipboardChange](#), [Sort's callback](#), [Critical](#), [Post/SendMessage](#), [Functions](#), [List of Windows Messages](#), [Threads](#)

## Examples

```
; Example: The following is a working script that
displays a summary of all top-level windows.
```

```
; For performance and memory conservation, call
RegisterCallback() only once for a given callback:
if not EnumAddress ; Fast-mode is okay because it
will be called only from this thread:
```

```
EnumAddress :=
```

```
RegisterCallback("EnumWindowsProc", "Fast")
```

```
DetectHiddenWindows On ; Due to fast-mode, this
setting will go into effect for the callback too.
```

```
; Pass control to EnumWindows(), which calls the
callback repeatedly:
```

```
DllCall("EnumWindows", Ptr, EnumAddress, Ptr, 0)
MsgBox %Output% ; Display the information
accumulated by the callback.
```

```

EnumWindowsProc(hwnd, lParam)
{
 global Output
 WinGetTitle, title, ahk_id %hwnd%
 WinGetClass, class, ahk_id %hwnd%
 if title
 Output .= "HWND: " . hwnd . "`tTitle: " .
title . "`tClass: " . class . "`n"
 return true ; Tell EnumWindows() to continue
until all windows have been enumerated.
}

```

; Example: The following is a working script that demonstrates how to subclass a GUI window by redirecting its WindowProc to a new WindowProc in the script. In this case, the background color of a text control is changed to a custom color.

```

TextBackgroundColor := 0xFFBBBB ; A custom color
in BGR format.
TextBackgroundBrush := DllCall("CreateSolidBrush",
 UInt, TextBackgroundColor)

```

```

Gui := GuiCreate()
Text := Gui.Add("Text",, "Here is some text that
is given`na custom background color.")

```

; 64-bit scripts must call SetWindowLongPtr
instead of SetWindowLong:

```

SetWindowLong := A_PtrSize=8 ? "SetWindowLongPtr"
: "SetWindowLong"

```

```

WindowProcNew := RegisterCallback("WindowProc", "")

```

```
; Specify "" to avoid fast-mode for subclassing.
, 4, Text.Hwnd) ; Must specify exact
ParamCount when EventInfo parameter is present.
WindowProcOld := DllCall(SetWindowLong, Ptr,
Gui.Hwnd, Int, -4 ; -4 is GWL_WNDPROC
, Ptr, WindowProcNew, Ptr) ; Return value must
be set to Ptr or UPtr vs. Int.
```

```
Gui.Show()
```

```
WindowProc(hwnd, uMsg, wParam, lParam)
{
 Critical
 global TextBackgroundColor,
 TextBackgroundBrush, WindowProcOld
 if (uMsg = 0x138 && lParam = A_EventInfo) ;
 0x138 is WM_CTLCOLORSTATIC.
 {
 DllCall("SetBkColor", Ptr, wParam, UInt,
TextBackgroundColor)
 return TextBackgroundBrush ; Return the
HBRUSH to notify the OS that we altered the HDC.
 }
 ; Otherwise (since above didn't return), pass
all unhandled events to the original WindowProc.
 return DllCall("CallWindowProc", Ptr,
WindowProcOld, Ptr, hwnd, UInt, uMsg, Ptr, wParam,
Ptr, lParam)
}
```

# StrGet

Copies a string to or from a memory address, optionally converting to or from a given code page.

```
OutputVar := StrGet(Address [, Length, Encoding =
None]
```

```
Function Example: string := StrGet(stringAddr,
10, "UTF-8")
```

## Parameters

### OutputVar

The name of the variable in which to store the read string.

### Address

The address at which the string will be written to/read from.

### Length

The maximum number of characters to read/write, including the null-terminator if required. See Return Value.

### Encoding

The source encoding for StrGet or target encoding for StrPut; for example, "UTF-8", "UTF-16" or "CP936". If *Address* and *Length* are not

specified, numeric identifiers must be prefixed with "CP". Specify an empty string or "CP0" to use the system default ANSI code page.

## Return Value

Invalid parameters cause an empty string to be returned, otherwise returns the requested string after performing any necessary conversion.

## Remarks

Note that the return value of StrGet are always in the [native encoding](#) of the current executable, whereas *Encoding* specifies the encoding of the string written to or read from the given *Address*. If no *Encoding* is specified, the string is simply copied without any conversion taking place.

If conversion between code pages is necessary, the required buffer size may differ from the size of the source *String*.

Scripts which are required to be compatible with AutoHotkey Basic can still use StrGet provided that [the appropriate script files](#) are installed in a [function library](#). These scripts can be found at the [AutoHotkey Community Forum](#).

## Related

[Script Compatibility](#), [StrPut](#), [StrPutVar](#), [FileEncoding](#), [VarSetCapacity](#)

## Examples

Either *Length* or *Encoding* may be specified directly after *Address*, but in those cases *Encoding* must be non-numeric:

```
strA := StrGet(addressA, "cp0") ; OK
strA := StrGet(addressA, length, 0) ; OK
strA := StrGet(addressA, 0) ; Error
```

# StrPut

Copies a string to or from a memory address, optionally converting to or from a given code page.

```
OutputVar := StrPut(String [, Address, Length,
Encoding = None])
```

```
Command Example: StrPut "Hello World!",
&string, 10, "UTF-8"
```

```
Function Example: ChrCount := StrPut("Hello
World!", &string, 10, "UTF-8")
```

## Parameters

### String

Any string. Numbers are also acceptable.

### Address

The address at which the string will be written to/read from.

### Length

The maximum number of characters to read/write, including the null-terminator if required. See Return Value.

### Encoding

The source encoding for StrGet or target encoding for StrPut; for

example, "UTF-8", "UTF-16" or "CP936". If *Address* and *Length* are not specified, numeric identifiers must be prefixed with "CP". Specify an empty string or "CP0" to use the system default ANSI code page.

## Return Value

For either function, invalid parameters cause an empty string to be returned.

StrPut returns the number of characters written, the required buffer size in characters if no *Address* was given, or 0 if an error occurred. If *Length* is less than the length of the source string, the function fails and returns 0. If *Length* is exactly the length of the source string, the string is not null-terminated; otherwise the returned count includes the null-terminator.

StrGet returns the requested string after performing any necessary conversion.

## Remarks

Note that the *String* parameter of StrPut and return value of StrGet are always in the [native encoding](#) of the current executable, whereas *Encoding* specifies the encoding of the string written to or read from the given *Address*. If no *Encoding* is specified, the string is simply measured or copied without any conversion taking place.

If conversion between code pages is necessary, the required buffer size may differ from the size of the source *String*.

Scripts which are required to be compatible with AutoHotkey Basic can still use

StrPut and StrGet provided that [the appropriate script files](#) are installed in a [function library](#). These scripts can be found at the [AutoHotkey Community Forum](#).

## Related

[Script Compatibility](#), [StrGet](#), [StrPutVar](#), [FileEncoding](#), [VarSetCapacity](#)

## Examples

Either *Length* or *Encoding* may be specified directly after *Address*, but in those cases *Encoding* must be non-numeric:

```
strA := StrGet(addressA, "cp0") ; OK
strA := StrGet(addressA, length, 0) ; OK
strA := StrGet(addressA, 0) ; Error
```

StrPut may be called once to calculate the required buffer size for a string in a particular encoding, then again to encode and write the string into the buffer. If you frequently use variables with StrPut, consider adding this function to your [library](#):

```
StrPutVar(string, ByRef var, encoding)
{
 ; Ensure capacity.
 VarSetCapacity(var, StrPut(string, encoding)
 ; StrPut returns char count, but
 VarSetCapacity needs bytes.
 * ((encoding="utf-16" || encoding="cp1200")
 ? 2 : 1))
}
```

```
; Copy or convert the string.
```

```
return StrPut(string, &var, encoding)
```

```
}
```

# VarSetCapacity()

Enlarges a variable's holding capacity or frees its memory. Normally, this is necessary only for unusual circumstances such as [DllCall](#).

```
OutputVar := VarSetCapacity(UnquotedVarName [, RequestedCapacity, FillByte])
```

```
Command Example: VarSetCapacity MyVar, 100
Function Example: GrantedCapacity :=
VarSetCapacity(MyVar, 100)
```

## Parameters

### OutputVar

The name of the variable in which to store granted capacity for the variable.

### UnquotedVarName

The name of the variable (*not in quotes*). For example:

`VarSetCapacity(MyVar, 1000)`. This can also be a dynamic variable such as `Array%i%` or a function's [ByRef](#) parameter.

### RequestedCapacity

If omitted, the variable's current capacity will be returned and its contents will not be altered. Otherwise, anything currently in the variable is lost

(the variable becomes blank).

Specify for *RequestedCapacity* the number of bytes that the variable should be able to hold after the adjustment. For Unicode strings, this should be the length times two. *RequestedCapacity* does not include the internal zero terminator. For example, specifying 1 would allow the variable to hold up to one byte in addition to its internal terminator. Note: the variable will auto-expand if the script assigns it a larger value later.

Since this function is often called simply to ensure the variable has a certain minimum capacity, for performance reasons, it shrinks the variable only when *RequestedCapacity* is 0. In other words, if the variable's capacity is already greater than *RequestedCapacity*, it will not be reduced (but the variable will still be made blank for consistency).

Therefore, to explicitly shrink a variable, first free its memory with

```
VarSetCapacity(Var, 0)
```

 and then use

```
VarSetCapacity(Var, NewCapacity) -- or simply let it auto-expand from zero as needed.
```

For performance reasons, freeing a variable whose previous capacity was less than 64 characters (128 bytes in Unicode builds) might have no effect because its memory is of a permanent type. In this case, the current capacity will be returned rather than 0.

For performance reasons, the memory of a variable whose capacity is less than 4096 bytes is not freed by storing an empty string in it (e.g. `Var :=`

`""`). However, `VarSetCapacity(Var, 0)` does free it.

Specify `-1` for *RequestedCapacity* to update the variable's internally-stored string length to the length of its current contents. This is useful in cases where the variable has been altered indirectly, such as by passing its [address](#) via [DllCall](#). In this mode, `VarSetCapacity()` returns the length in bytes rather than the capacity.

### FillByte

This parameter is normally omitted, in which case the memory of the target variable is not initialized (instead, the variable is simply made blank as described above). Otherwise, specify a number between 0 and 255. Each byte in the target variable's memory area (its current capacity, which might be greater than *RequestedCapacity*) is set to that number. Zero is by far the most common value, which is useful in cases where the variable will hold raw binary data such as a [DllCall structure](#).

### Remarks

In addition to its uses described at [DllCall](#), this function can also be used to enhance performance when building a string by means of gradual concatenation. This is because multiple automatic resizings can be avoided when you have some idea of what the string's final length will be. In such a case, *RequestedCapacity* need not be accurate: if the capacity is too small, performance is still improved and the variable will begin auto-expanding when the capacity has been exhausted. If the capacity is too large, some of the memory

is wasted, but only temporarily because all the memory can be freed after the operation by means of `VarSetCapacity(Var, 0)` or `Var := ""`.

## Related

[DllCall](#), [NumPut](#), [NumGet](#)

## Example

```
; Optimize by ensuring MyVar has plenty of space to work with.
```

```
VarSetCapacity(MyVar, 10240000) ; ~10 MB
```

```
Loop
```

```
{
```

```
 ...
```

```
 MyVar .= StringToConcatenate
```

```
 ...
```

```
}
```

```
; Calculate required buffer space for a string.
```

```
max_chars := 500
```

```
max_bytes := max_chars * 2
```

```
Loop 2
```

```
{
```

```
 ; Allocate space for use with DllCall.
```

```
 VarSetCapacity(buf, max_bytes)
```

```
 if A_Index = 1
```

```
 ; Alter the variable indirectly via DllCall.
```

```
 DllCall("wsprintf", "ptr", &buf, "str",
"0x`%08x", "uint", 4919)
```

```

else
 ; Use "str" to update the length
 automatically:
 DllCall("wsprintf", "str", buf, "str",
 "0x`%08x", "uint", 4919)

 ; Concatenate a string to demonstrate why the
 length needs to be updated:
 wrong_str := buf . "<end>"
 wrong_len := StrLen(buf)

 ; Update the variable's length.
 VarSetCapacity(buf, -1)

 right_str := buf . "<end>"
 right_len := StrLen(buf)

 MsgBox,
 (
 Before updating
 String: %wrong_str%
 Length: %wrong_len%

 After updating
 String: %right_str%
 Length: %right_len%
)
}

```

# WinApi

Creates a script function for many windows dlls functions automatically.

```
OutputVar := Function([Arg1, Arg2, ...])
```

```
Function Example: cmd := GetCommandLine()
```

## Parameters

### OutputVar

The name of the variable in which to store the actual return value of a function same as in [DllCall](#) or [DynaCall](#). If the function is of a type that does not return a value, the result is an undefined integer. If the function cannot be called due to an [error](#), the OutputVar is set blank (an empty string) and ErrorLevel is set.

### Function

The name of dll function to call. See example.

### Arg1, Arg2, ... (optional)

Parameters to be passed to function.

All parameters are optional, when omitted "" will be used for AStr, WStr and Str and 0 otherwise.

## General Remarks

Some functions are in conflict with ahk functions, those are appended an underscore: Sleep\_, SendInput\_, SendMessage\_, PostMessage\_, BlockInput\_, GetKeyState\_, SetTimer\_, Send\_, Shutdown\_.

Parameter \* or P is not used in WinApi, instead such parameters are of type PTR, use `getvar(var:=0)` if you like to store the value in variable directly. The variable is of type Int64 by default, to convert to desired type use [Cast](#) or [ToChar](#), [ToShort](#), [ToInt](#), [ToUChar](#), [ToUShort](#), [ToUInt](#), this is mainly required to convert Unsigned value to Signed.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#)

## Examples

```
MsgBox % LoadLibrary(A_AhkPath)

GetCommandLine,cmd
MsgBox % cmd

StrToIntEx("-2147483648",0,getvar(var:=0))
MsgBox % ToInt(var)

OnMessage(0x999,"0x999")
SendMessage_(A_ScriptHwnd,0x999) ; same as
DllCall("user32\SendMessage","PTR",A_ScriptHwnd,"U
INT",0x999,"PTR",0,"PTR",0,"PTR")
ExitApp
0x999(w,l,m,h){
 MsgBox % w "`n" l "`n" m "`n" h
```

```
return 1
}
```

## Definitions

Each table below has 3 columns.

**First** column contains ahk definition using following types (last parameter is always the return type).

| <b>DllCall</b>                        | <b>DynaCall equivalent</b> |
|---------------------------------------|----------------------------|
| Int                                   | i                          |
| Str                                   | s                          |
| AStr                                  | a                          |
| WStr                                  | w                          |
| Short                                 | h                          |
| Char                                  | c                          |
| Float                                 | f                          |
| Double                                | d                          |
| PTR                                   | t                          |
| Int64                                 | i6                         |
| Int (64-bit) /<br>Short (32-bit)      | v                          |
| UInt (64-bit) /<br>Ushort (32-bit)    | x                          |
| Short (Unicode)<br>/ Char (Ansi)      | y                          |
| Ushort<br>(Unicode) /<br>Uchar (Ansi) | z                          |

**Second** column shows windows return type of function.

**Third** column shows the function and original parameter definition.

```
Advapi32.dll, Comctl32.dll, Comdlg32.dll,
Crypt32.dll, Gdi32.dll, Gdiplus.dll, Glu32.dll,
Hid.dll, Kernel32.dll, Ole32.dll, Oleacc.dll,
OleAut32.dll, Opengl32.dll, Rasapi32.dll,
Rasdmg.dll, Rasman.dll, Shell32.dll,
Shlwapi.dll, Tapi32.dll, User32.dll,
Userenv.dll, UxTheme.dll, Version.dll,
Winhttp.dll, Wininet.dll, Winmm.dll, Ws2_32.dll
```

# Advapi32.dll

|               |      |                                                                                                                                                                                                                                                                                                                               |
|---------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si            | BOOL | <a href="#">AbortSystemShutdown</a> (_In_opt_ LPTSTR                                                                                                                                                                                                                                                                          |
| ai            | BOOL | <a href="#">AbortSystemShutdownA</a> (_In_opt_ LPST                                                                                                                                                                                                                                                                           |
| wi            | BOOL | <a href="#">AbortSystemShutdownW</a> (_In_opt_ LPW                                                                                                                                                                                                                                                                            |
| ttuitttti     | BOOL | <a href="#">AccessCheck</a> (_In_ PSECURITY_DESC<br>pSecurityDescriptor, _In_ HANDLE Cli<br>DesiredAccess, _In_ PGENERIC_MAPI<br>_Out_opt_ PPRIVILEGE_SET Privilege<br>PrivilegeSetLength, _Out_ LPDWORD (L<br>LPBOOL AccessStatus)                                                                                           |
| stststuititti | BOOL | <a href="#">AccessCheckAndAuditAlarm</a> (_In_ LPC<br>_In_opt_ LPVOID HandleId, _In_ LPTS<br>_In_opt_ LPTSTR ObjectName, _In_<br>PSECURITY_DESCRIPTOR SecurityD<br>DesiredAccess, _In_ PGENERIC_MAPI<br>_In_ BOOL ObjectCreation, _Out_ LPD<br>_Out_ LPBOOL AccessStatus, _Out_ LP<br>pfGenerateOnClose)                      |
| ataatuititti  | BOOL | <a href="#">AccessCheckAndAuditAlarmA</a> (_In_ LP<br>_In_opt_ LPVOID HandleId, _In_ LPST<br>_In_opt_ LPSTR ObjectName, _In_<br>PSECURITY_DESCRIPTOR SecurityD<br>DesiredAccess, _In_ PGENERIC_MAPI<br>_In_ BOOL ObjectCreation, _Out_ LPD<br>_Out_ LPBOOL AccessStatus, _Out_ LP<br>pfGenerateOnClose)                       |
| wtwwtuititti  | BOOL | <a href="#">AccessCheckAndAuditAlarmW</a> (_In_ LP<br>SubsystemName, _In_opt_ LPVOID Ha<br>ObjectName, _In_opt_ LPWSTR O<br>PSECURITY_DESCRIPTOR SecurityD<br>DesiredAccess, _In_ PGENERIC_MAPI<br>_In_ BOOL ObjectCreation, _Out_ LPD<br>_Out_ LPBOOL AccessStatus, _Out_ LP<br>pfGenerateOnClose)                           |
| tttuititttti  | BOOL | <a href="#">AccessCheckByType</a> (_In_ PSECURITY<br>pSecurityDescriptor, _In_opt_ PSID Pri<br>HANDLE ClientToken, _In_ DWORD I<br>_Inout_opt_ POBJECT_TYPE_LIST Ob<br>DWORD ObjectTypeIdListLength, _In_ P<br>GenericMapping, _Out_opt_ PPRIVILE<br>_Inout_ LPDWORD PrivilegeSetLength<br>GrantedAccess, _Out_ LPBOOL Access |
|               |      | <a href="#">AccessCheckByTypeAndAuditAlarm</a> (I<br>SubsystemName, _In_ LPVOID Handle                                                                                                                                                                                                                                        |

|                     |      |                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stssttuiuuituititti | BOOL | ObjectTypeName, _In_opt_ LPCTSTR (PSECURITY_DESCRIPTOR pSecurity, PSID PrincipalSelfSid, _In_ DWORD D AUDIT_EVENT_TYPE AuditType, _In_ _Inout_opt_ POBJECT_TYPE_LIST Ob DWORD ObjectTypeListLength, _In_ P GenericMapping, _In_ BOOL ObjectCre LPDWORD GrantedAccess, _Out_ LPB _Out_ LPBOOL pfGenerateOnClose)                                                                                      |
| ataattuiuuituititti | BOOL | <a href="#">AccessCheckByTypeAndAuditAlarmA</a> (SubsystemName, _In_ LPVOID Handle, ObjectTypeName, _In_opt_ LPCSTR Ob PSECURITY_DESCRIPTOR pSecurity, PSID PrincipalSelfSid, _In_ DWORD D AUDIT_EVENT_TYPE AuditType, _In_ _Inout_opt_ POBJECT_TYPE_LIST Ob DWORD ObjectTypeListLength, _In_ P GenericMapping, _In_ BOOL ObjectCre LPDWORD GrantedAccess, _Out_ LPB _Out_ LPBOOL pfGenerateOnClose) |
| wtwwtuiuuituititti  | BOOL | <a href="#">AccessCheckByTypeAndAuditAlarmW</a> (SubsystemName, _In_ LPVOID Handle, ObjectTypeName, _In_opt_ LPCWSTR PSECURITY_DESCRIPTOR pSecurity, PSID PrincipalSelfSid, _In_ DWORD D AUDIT_EVENT_TYPE AuditType, _In_ _Inout_opt_ POBJECT_TYPE_LIST Ob DWORD ObjectTypeListLength, _In_ P GenericMapping, _In_ BOOL ObjectCre LPDWORD GrantedAccess, _Out_ LPB _Out_ LPBOOL pfGenerateOnClose)   |
| tttuitititti        | BOOL | <a href="#">AccessCheckByTypeResultList</a> (_In_ PSECURITY_DESCRIPTOR pSecurity, PSID PrincipalSelfSid, _In_ HANDLE C DWORD DesiredAccess, _Inout_opt_ P ObjectTypelist, _In_ DWORD ObjectTy PGENERIC_MAPPING GenericMappin PPRIVILEGE_SET PrivilegeSet, _Inout_ PrivilegeSetLength, _Out_ LPDWORD ( _Out_ LPDWORD AccessStatusList)                                                                |
| stssttuiuuituititti | BOOL | <a href="#">AccessCheckByTypeResultListAndAudit</a> (SubsystemName, _In_ LPVOID Handle, ObjectTypeName, _In_opt_ LPCTSTR (PSECURITY_DESCRIPTOR pSecurity, PSID PrincipalSelfSid, _In_ DWORD D AUDIT_EVENT_TYPE AuditType, _In_ _Inout_opt_ POBJECT_TYPE_LIST Ob DWORD ObjectTypeListLength, _In_ P GenericMapping, _In_ BOOL ObjectCre LPDWORD GrantedAccess, _Out_ LPD                              |

|                      |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |      | AccessStatusList, _Out_ LPBOOL pfGe                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| ataattuiiuituititti  | BOOL | <a href="#">AccessCheckByTypeResultListAndAud</a><br>SubsystemName, _In_ LPVOID Handle<br>ObjectTypeName, _In_opt_ LPCSTR Ob<br>PSECURITY_DESCRIPTOR pSecurity<br>PSID PrincipalSelfSid, _In_ DWORD D<br>AUDIT_EVENT_TYPE AuditType, _In<br>_Inout_opt_ POBJECT_TYPE_LIST Ob<br>DWORD ObjectTypeListLength, _In_ P<br>GenericMapping, _In_ BOOL ObjectCre<br>LPDWORD GrantedAccess, _Out_ LPD<br>AccessStatusList, _Out_ LPBOOL pfGe                                   |
| sttssttuiiuituititti | BOOL | <a href="#">AccessCheckByTypeResultListAndAud</a><br>LPCTSTR SubsystemName, _In_ LPVC<br>HANDLE ClientToken, _In_ LPCTSTR<br>_In_opt_ LPCTSTR ObjectName, _In_<br>PSECURITY_DESCRIPTOR pSecurity<br>PSID PrincipalSelfSid, _In_ DWORD D<br>AUDIT_EVENT_TYPE AuditType, _In<br>_Inout_opt_ POBJECT_TYPE_LIST Ob<br>DWORD ObjectTypeListLength, _In_ P<br>GenericMapping, _In_ BOOL ObjectCre<br>LPDWORD GrantedAccess, _Out_ LPD<br>AccessStatusList, _Out_ LPBOOL pfGe |
| attaattuiiuituititti | BOOL | <a href="#">AccessCheckByTypeResultListAndAud</a><br>LPCSTR SubsystemName, _In_ LPVOI<br>HANDLE ClientToken, _In_ LPCSTR C<br>_In_opt_ LPCSTR ObjectName, _In_<br>PSECURITY_DESCRIPTOR pSecurity<br>PSID PrincipalSelfSid, _In_ DWORD D<br>AUDIT_EVENT_TYPE AuditType, _In<br>_Inout_opt_ POBJECT_TYPE_LIST Ob<br>DWORD ObjectTypeListLength, _In_ P<br>GenericMapping, _In_ BOOL ObjectCre<br>LPDWORD GrantedAccess, _Out_ LPD<br>AccessStatusList, _Out_ LPBOOL pfGe |
| wttwttuiiuituititti  | BOOL | <a href="#">AccessCheckByTypeResultListAndAud</a><br>LPCWSTR SubsystemName, _In_ LPVC<br>HANDLE ClientToken, _In_ LPCWSTR<br>_In_opt_ LPCWSTR ObjectName, _In_<br>PSECURITY_DESCRIPTOR pSecurity<br>PSID PrincipalSelfSid, _In_ DWORD D<br>AUDIT_EVENT_TYPE AuditType, _In<br>_Inout_opt_ POBJECT_TYPE_LIST Ob<br>DWORD ObjectTypeListLength, _In_ P<br>GenericMapping, _In_ BOOL ObjectCre<br>LPDWORD GrantedAccess, _Out_ LPD<br>AccessStatusList, _Out_ LPBOOL pfGe |
|                      |      | <a href="#">AccessCheckByTypeResultListAndAud</a><br>LPCWSTR SubsystemName, _In_ LPVC                                                                                                                                                                                                                                                                                                                                                                                  |

|                     |      |                                                                                                                                                                                                                                                                                                                                               |
|---------------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wtwwttuiiuituititti | BOOL | LPCWSTR ObjectTypeName, _In_opt_<br>_In_ PSECURITY_DESCRIPTOR pSec<br>_In_opt_ PSID PrincipalSelfSid, _In_ D<br>_In_ AUDIT_EVENT_TYPE AuditType<br>_Inout_opt_ POBJECT_TYPE_LIST Ob<br>DWORD ObjectTypeIdListLength, _In_ P<br>GenericMapping, _In_ BOOL ObjectCre<br>LPDWORD GrantedAccess, _Out_ LPD<br>AccessStatusList, _Out_ LPBOOL pfGe |
| tuiuiti             | BOOL | AddAccessAllowedAce(_Inout_ PACL p<br>dwAceRevision, _In_ DWORD AccessM                                                                                                                                                                                                                                                                       |
| tuiuiuiti           | BOOL | AddAccessAllowedAceEx(_Inout_ PAC<br>dwAceRevision, _In_ DWORD AceFlag<br>AccessMask, _In_ PSID pSid)                                                                                                                                                                                                                                         |
| tuiuiuitti          | BOOL | AddAccessAllowedObjectAce(_Inout_ P<br>DWORD dwAceRevision, _In_ DWOR<br>DWORD AccessMask, _In_opt_ GUID<br>_In_opt_ GUID *InheritedObjectTypeGu                                                                                                                                                                                              |
| tuiuiti             | BOOL | AddAccessDeniedAce(_Inout_ PACL pA<br>dwAceRevision, _In_ DWORD AccessM                                                                                                                                                                                                                                                                       |
| tuiuiuiti           | BOOL | AddAccessDeniedAceEx(_Inout_ PACL<br>dwAceRevision, _In_ DWORD AceFlag<br>AccessMask, _In_ PSID pSid)                                                                                                                                                                                                                                         |
| tuiuiuitti          | BOOL | AddAccessDeniedObjectAce(_Inout_ PA<br>dwAceRevision, _In_ DWORD AceFlag<br>AccessMask, _In_opt_ GUID *ObjectTy<br>*InheritedObjectTypeGuid, _In_ PSID p                                                                                                                                                                                      |
| tuiuitui            | BOOL | AddAce(_Inout_ PACL pAcl, _In_ DW<br>_In_ DWORD dwStartingAceIndex, _In<br>_In_ DWORD nAceListLength)                                                                                                                                                                                                                                         |
| tuiuitiii           | BOOL | AddAuditAccessAce(_Inout_ PACL pAc<br>dwAceRevision, _In_ DWORD dwAcce<br>_In_ BOOL bAuditSuccess, _In_ BOOL                                                                                                                                                                                                                                  |
| tuiuiuitiii         | BOOL | AddAuditAccessAceEx(_Inout_ PACL p<br>dwAceRevision, _In_ DWORD AceFlag<br>dwAccessMask, _In_ PSID pSid, _In_ B<br>_In_ BOOL bAuditFailure)                                                                                                                                                                                                   |
| tuiuiuitttiii       | BOOL | AddAuditAccessObjectAce(_Inout_ PAC<br>dwAceRevision, _In_ DWORD AceFlag<br>AccessMask, _In_opt_ GUID *ObjectTy<br>*InheritedObjectTypeGuid, _In_ PSID p<br>bAuditSuccess, _In_ BOOL bAuditFailu                                                                                                                                              |
| tuiuiucuitwti       | BOOL | AddConditionalAce(_Inout_ PACL pAcl<br>dwAceRevision, _In_ DWORD AceFlag<br>AceType, _In_ DWORD AccessMask, _<br>PWCHAR ConditionStr, _Out_ DWOR                                                                                                                                                                                              |
|                     |      |                                                                                                                                                                                                                                                                                                                                               |

|                     |         |                                                                                                                                                                                                                                                                           |
|---------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiuti            | BOOL    | AddMandatoryAce(_Inout_ PACL pAcl, dwAceRevision, _In_ DWORD AceFlag MandatoryPolicy, _In_ PSID pLabelSid)                                                                                                                                                                |
| wtui                | DWORD   | AddUsersToEncryptedFile(_In_ LPCWSTR PENCRYPTION_CERTIFICATE_LIST                                                                                                                                                                                                         |
| tituitti            | BOOL    | AdjustTokenGroups(_In_ HANDLE Tok ResetToDefault, _In_opt_ PTOKEN_GR DWORD BufferLength, _Out_opt_ PTO PreviousState, _Out_opt_ PDWORD Re                                                                                                                                 |
| tituitti            | BOOL    | AdjustTokenPrivileges(_In_ HANDLE T BOOL DisableAllPrivileges, _In_opt_ P NewState, _In_ DWORD BufferLength, PTOKEN_PRIVILEGES PreviousState, ReturnLength)                                                                                                               |
| tucuiuiuiuiuiuiuiui | BOOL    | AllocateAndInitializeSid(_In_ PSID_IDENTIFIER_AUTHORITY pIde BYTE nSubAuthorityCount, _In_ DWO _In_ DWORD dwSubAuthority1, _In_ D dwSubAuthority2, _In_ DWORD dwSub DWORD dwSubAuthority4, _In_ DWO _In_ DWORD dwSubAuthority6, _In_ D dwSubAuthority7, _Out_ PSID *pSid) |
| ti                  | BOOL    | AllocateLocallyUniqueId(_Out_ PLUID                                                                                                                                                                                                                                       |
| uiuii               | BOOL    | AreAllAccessesGranted(_In_ DWORD C DWORD DesiredAccess)                                                                                                                                                                                                                   |
| uiuii               | BOOL    | AreAnyAccessesGranted(_In_ DWORD D WORD DesiredAccess)                                                                                                                                                                                                                    |
| ttuituc             | BOOLEAN | AuditComputeEffectivePolicyBySid(_In const GUID *pSubCategoryGuids, _In_ _Out_ PAUDIT_POLICY_INFORMATION                                                                                                                                                                  |
| ttuituc             | BOOLEAN | AuditComputeEffectivePolicyByToken(hTokenHandle, _In_ const GUID *pSub( ULONG PolicyCount, _Out_ PAUDIT_POLICY_INFORMATION *p                                                                                                                                             |
| ttuc                | BOOLEAN | AuditEnumerateCategories(_Out_ GUID **ppAuditCategoriesArray, _Out_ PULC                                                                                                                                                                                                  |
| tuc                 | BOOLEAN | AuditEnumeratePerUserPolicy(_Out_ PPOLICY_AUDIT_SID_ARRAY *ppA                                                                                                                                                                                                            |
| tucttuc             | BOOLEAN | AuditEnumerateSubCategories(_In_ con *pAuditCategoryGuid, _In_ BOOLEAN bRetrieveAllSubCategories, _Out_ GUI **ppAuditSubCategoriesArray, _Out_ P pCountReturned)                                                                                                          |
| ti                  | VOID    | AuditFree(_In_ PVOID Buffer)                                                                                                                                                                                                                                              |
|                     |         | AuditLookupCategoryGuidFromCategory                                                                                                                                                                                                                                       |

|         |         |                                                                                                                                   |
|---------|---------|-----------------------------------------------------------------------------------------------------------------------------------|
| uituc   | BOOLEAN | POLICY_AUDIT_EVENT_TYPE AuditCategoryGuid *pAuditCategoryGuid)                                                                    |
| ttuc    | BOOLEAN | AuditLookupCategoryIdFromCategoryGuid *pAuditCategoryGuid, _Out_ PPOLICY_AUDIT_EVENT_TYPE pAuditCategoryGuid)                     |
| tsuc    | BOOLEAN | AuditLookupCategoryName(_In_ const GUID *pAuditCategoryGuid, _Out_ PTSTR *pCategoryName)                                          |
| tauc    | BOOLEAN | AuditLookupCategoryNameA(_In_ const GUID *pAuditCategoryGuid, _Out_ PTSTR *pCategoryName)                                         |
| twuc    | BOOLEAN | AuditLookupCategoryNameW(_In_ const GUID *pAuditCategoryGuid, _Out_ PTSTR *pCategoryName)                                         |
| tsuc    | BOOLEAN | AuditLookupSubCategoryName(_In_ const GUID *pAuditSubCategoryGuid, _Out_ PTSTR *pSubCategoryName)                                 |
| tauc    | BOOLEAN | AuditLookupSubCategoryNameA(_In_ const GUID *pAuditSubCategoryGuid, _Out_ PTSTR *pSubCategoryName)                                |
| twuc    | BOOLEAN | AuditLookupSubCategoryNameW(_In_ const GUID *pAuditSubCategoryGuid, _Out_ PTSTR *pSubCategoryName)                                |
| wtuc    | BOOLEAN | AuditQueryGlobalSacl(_In_ PCWSTR ObjectName PACL *Acl)                                                                            |
| atuc    | BOOLEAN | AuditQueryGlobalSaclA(_In_ PCWSTR ObjectName _Out_ PACL *Acl)                                                                     |
| wtuc    | BOOLEAN | AuditQueryGlobalSaclW(_In_ PCWSTR ObjectName _Out_ PACL *Acl)                                                                     |
| ttuituc | BOOLEAN | AuditQueryPerUserPolicy(_In_ const PSID *pSubCategoryGuids, _In_ ULONG PolicyIndex PAUDIT_POLICY_INFORMATION *pPolicyInformation) |
| uituc   | BOOLEAN | AuditQuerySecurity(_In_ SECURITY_INFORMATION SecurityInformation, _Out_ PSECURITY_DESCRIPTOR *ppSecurityDescriptor)               |
| tuituc  | BOOLEAN | AuditQuerySystemPolicy(_In_ const GUID *pSubCategoryGuids, _In_ ULONG PolicyIndex PAUDIT_POLICY_INFORMATION *pPolicyInformation)  |
| wtuc    | BOOLEAN | AuditSetGlobalSacl(_In_ PCWSTR ObjectName PACL Acl)                                                                               |
| atuc    | BOOLEAN | AuditSetGlobalSaclA(_In_ PCWSTR ObjectName _In_opt_ PACL Acl)                                                                     |
| wtuc    | BOOLEAN | AuditSetGlobalSaclW(_In_ PCWSTR ObjectName _In_opt_ PACL Acl)                                                                     |
| ttuiuc  | BOOLEAN | AuditSetPerUserPolicy(_In_ const PSID *pSubCategoryGuids, _In_ ULONG PolicyIndex PCAUDIT_POLICY_INFORMATION *pPolicyInformation)  |

|              |         |                                                                                                                                                                                                                                                                          |
|--------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |         | ULONG PolicyCount)                                                                                                                                                                                                                                                       |
| uituc        | BOOLEAN | AuditSetSecurity(_In_ SECURITY_INFORMATION SecurityInformation, _In_ PSECURITY_DESCRIPTOR pSecurityDescriptor)                                                                                                                                                           |
| tuiuc        | BOOLEAN | AuditSetSystemPolicy(_In_ PCAUDIT_POLICY_INFORMATION pPolicyInformation, ULONG PolicyCount)                                                                                                                                                                              |
| tsi          | BOOL    | BackupEventLog(_In_ HANDLE hEventLog, LPCTSTR lpBackupFileName)                                                                                                                                                                                                          |
| tai          | BOOL    | BackupEventLogA(_In_ HANDLE hEventLog, LPCTSTR lpBackupFileName)                                                                                                                                                                                                         |
| twi          | BOOL    | BackupEventLogW(_In_ HANDLE hEventLog, LPCTSTR lpBackupFileName)                                                                                                                                                                                                         |
| tsuiuiiii    | VOID    | BuildExplicitAccessWithName(_Inout_ PEXPLICIT_ACCESS pExplicitAccess, _In_opt_ LPCTSTR pTrusteeName, DWORD AccessPermissions, _In_ ACCESS_MODE AccessMode, _In_ DWORD InheritanceFlags)                                                                                  |
| tauiuiiii    | VOID    | BuildExplicitAccessWithNameA(_Inout_ PEXPLICIT_ACCESS pExplicitAccess, _In_opt_ LPSTR pTrusteeName, DWORD AccessPermissions, _In_ ACCESS_MODE AccessMode, _In_ DWORD InheritanceFlags)                                                                                   |
| twuiuiiii    | VOID    | BuildExplicitAccessWithNameW(_Inout_ PEXPLICIT_ACCESS pExplicitAccess, _In_opt_ LPWSTR pTrusteeName, DWORD AccessPermissions, _In_ ACCESS_MODE AccessMode, _In_ DWORD InheritanceFlags)                                                                                  |
| ttuituitttui | DWORD   | BuildSecurityDescriptor(_In_opt_ PTRUSTEE pGroup, _In_ ULONG cCountOfAccessEntries, _In_opt_ PEXPLICIT_ACCESS pListOfAccessEntries, _In_opt_ PSECURITY_DESCRIPTOR pOldSD, _Out_ PULONG pSizeNewSD, PSECURITY_DESCRIPTOR *pNewSD)                                         |
| ttuituitttui | DWORD   | BuildSecurityDescriptorA(_In_opt_ PTRUSTEE pGroup, _In_ ULONG cCountOfAccessEntries, _In_opt_ PEXPLICIT_ACCESS pListOfAccessEntries, _In_ ULONG cCountOfAuditEntries, _In_opt_ PSECURITY_DESCRIPTOR pOldSD, _Out_ PULONG pSizeNewSD, _Out_ PSECURITY_DESCRIPTOR *pNewSD) |
| ttuituitttui | DWORD   | BuildSecurityDescriptorW(_In_opt_ PTRUSTEE pGroup, _In_ ULONG cCountOfAccessEntries, _In_opt_ PEXPLICIT_ACCESS pListOfAccessEntries, _In_ ULONG cCountOfAuditEntries, _In_opt_ PSECURITY_DESCRIPTOR pOldSD, _Out_ PULONG pSizeNewSD, _Out_ PSECURITY_DESCRIPTOR *pNewSD) |

|               |      |                                                                                                                                                                                                                                                                     |
|---------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsi           | VOID | BuildTrusteeWithName(_Inout_ PTRUS<br>LPTSTR pName)                                                                                                                                                                                                                 |
| tai           | VOID | BuildTrusteeWithNameA(_Inout_ PTRU<br>_In_opt_ LPSTR pName)                                                                                                                                                                                                         |
| twi           | VOID | BuildTrusteeWithNameW(_Inout_ PTRU<br>_In_opt_ LPWSTR pName)                                                                                                                                                                                                        |
| ttuissssi     | VOID | BuildTrusteeWithObjectsAndName(_Ino<br>pTrustee, _In_opt_ POBJECTS_AND_N<br>_In_opt_ SE_OBJECT_TYPE ObjectTy<br>ObjectName, _In_opt_ LPTSTR Inl<br>_In_opt_ LPTSTR Name)                                                                                            |
| ttuiaaai      | VOID | BuildTrusteeWithObjectsAndNameA(_I<br>pTrustee, _In_opt_ POBJECTS_AND_N<br>_In_opt_ SE_OBJECT_TYPE ObjectTy<br>ObjectName, _In_opt_ LPSTR Inhe<br>_In_opt_ LPSTR Name)                                                                                              |
| ttuiwwwi      | VOID | BuildTrusteeWithObjectsAndNameW(_I<br>pTrustee, _In_opt_ POBJECTS_AND_N<br>_In_opt_ SE_OBJECT_TYPE ObjectTy<br>ObjectName, _In_opt_ LPWSTR<br>InheritedObjectName, _In_opt_ LP                                                                                      |
| tttiti        | VOID | BuildTrusteeWithObjectsAndSid(_Inout<br>_In_opt_ POBJECTS_AND_SID pObjS<br>*pObjectGuid, _In_opt_ GUID *pInheri<br>PSID pSid)                                                                                                                                       |
| tttiti        | VOID | BuildTrusteeWithObjectsAndSidA(_Ino<br>pTrustee, _In_opt_ POBJECTS_AND_S<br>GUID *pObjectGuid, _In_opt_ GUID *p<br>_In_opt_ PSID pSid)                                                                                                                              |
| tttiti        | VOID | BuildTrusteeWithObjectsAndSidW(_Ino<br>pTrustee, _In_opt_ POBJECTS_AND_S<br>GUID *pObjectGuid, _In_opt_ GUID *p<br>_In_opt_ PSID pSid)                                                                                                                              |
| titi          | VOID | BuildTrusteeWithSid(_Inout_ PTRUSTE<br>PSID pSid)                                                                                                                                                                                                                   |
| titi          | VOID | BuildTrusteeWithSidA(_Inout_ PTRUST<br>PSID pSid)                                                                                                                                                                                                                   |
| titi          | VOID | BuildTrusteeWithSidW(_Inout_ PTRUS<br>PSID pSid)                                                                                                                                                                                                                    |
| tuiuiisstsssi | BOOL | ChangeServiceConfig(_In_ SC_HANDL<br>DWORD dwServiceType, _In_ DWOR<br>DWORD dwErrorControl, _In_opt_ LPC<br>lpBinaryPathName, _In_opt_ LPCTSTR<br>_Out_opt_ LPDWORD lpdwTagId, _In_<br>lpDependencies, _In_opt_ LPCTSTR lpS<br>_In_opt_ LPCTSTR lpPassword, _In_op |

|                 |      |                                                                                                                                                                                                                                     |
|-----------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |      | lpDisplayName)                                                                                                                                                                                                                      |
| tuiti           | BOOL | ChangeServiceConfig2(_In_ SC_HANDLE hService, _In_ DWORD dwInfoLevel, _In_opt_ LPVOID lpInfo)                                                                                                                                       |
| tuiti           | BOOL | ChangeServiceConfig2A(_In_ SC_HANDLE hService, _In_ DWORD dwInfoLevel, _In_opt_ LPVOID lpInfo)                                                                                                                                      |
| tuiti           | BOOL | ChangeServiceConfig2W(_In_ SC_HANDLE hService, _In_ DWORD dwInfoLevel, _In_opt_ LPVOID lpInfo)                                                                                                                                      |
| tuiuiuiaataaaai | BOOL | ChangeServiceConfigA(_In_ SC_HANDLE hService, _In_ DWORD dwServiceType, _In_ DWORD dwErrorControl, _In_opt_ LPCWSTR lpBinaryPathName, _In_opt_ LPCWSTR lpDependencies, _In_opt_ LPCWSTR lpPassword, _In_opt_ LPCWSTR lpDisplayName) |
| tuiuiuwwtwwwwi  | BOOL | ChangeServiceConfigW(_In_ SC_HANDLE hService, _In_ DWORD dwServiceType, _In_ DWORD dwErrorControl, _In_opt_ LPCWSTR lpBinaryPathName, _In_opt_ LPCWSTR lpDependencies, _In_opt_ LPCWSTR lpPassword, _In_opt_ LPCWSTR lpDisplayName) |
| ttti            | BOOL | CheckTokenMembership(_In_opt_ HANDLE hToken, _In_ PSID SidToCheck, _Out_ PBOOL pIsMember)                                                                                                                                           |
| tsi             | BOOL | ClearEventLog(_In_ HANDLE hEventLog, _In_opt_ LPCTSTR lpBackupFileName)                                                                                                                                                             |
| tai             | BOOL | ClearEventLogA(_In_ HANDLE hEventLog, _In_opt_ LPCTSTR lpBackupFileName)                                                                                                                                                            |
| twi             | BOOL | ClearEventLogW(_In_ HANDLE hEventLog, _In_opt_ LPCTSTR lpBackupFileName)                                                                                                                                                            |
| ti              | VOID | CloseEncryptedFileRaw(_In_ PVOID pFile)                                                                                                                                                                                             |
| ti              | BOOL | CloseEventLog(_Inout_ HANDLE hEventLog)                                                                                                                                                                                             |
| ti              | BOOL | CloseServiceHandle(_In_ SC_HANDLE hService)                                                                                                                                                                                         |
| ti              | VOID | CloseThreadWaitChainSession(_In_ HANDLE hThread)                                                                                                                                                                                    |
| tuiti           | BOOL | ControlService(_In_ SC_HANDLE hService, _In_ DWORD dwControl, _Out_ LPSERVICE_STATUS lpServiceStatus)                                                                                                                               |
| tuiuiti         | BOOL | ControlServiceEx(_In_ SC_HANDLE hService, _In_ DWORD dwControl, _In_ DWORD dwInfoLevel, _In_ LPVOID lpControlParams)                                                                                                                |
| tuiuiti         | BOOL | ControlServiceExA(_In_ SC_HANDLE hService, _In_ DWORD dwControl, _In_ DWORD dwInfoLevel, _In_ LPVOID lpControlParams)                                                                                                               |
|                 |      |                                                                                                                                                                                                                                     |

|         |      |                                                                                                                                                                                                                                                              |
|---------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiti | BOOL | ControlServiceExW(_In_ SC_HANDLE dwControl, _In_ DWORD dwInfoLevel, pControlParams)                                                                                                                                                                          |
| tuiuiti | BOOL | ConvertSecurityDescriptorToStringSecurityDescriptor(PSECURITY_DESCRIPTOR SecurityDescriptor, RequestedStringSDRevision, _In_ SECURITY_INFORMATION SecurityInformation, _Out_ LPTSTR *StringSecurityDescriptor, _Out_ PULONG StringSecurityDescriptorLength)  |
| tuiuiti | BOOL | ConvertSecurityDescriptorToStringSecurityDescriptorA(PSECURITY_DESCRIPTOR SecurityDescriptor, RequestedStringSDRevision, _In_ SECURITY_INFORMATION SecurityInformation, _Out_ LPSTR *StringSecurityDescriptor, _Out_ PULONG StringSecurityDescriptorLength)  |
| tuiuiti | BOOL | ConvertSecurityDescriptorToStringSecurityDescriptorW(PSECURITY_DESCRIPTOR SecurityDescriptor, RequestedStringSDRevision, _In_ SECURITY_INFORMATION SecurityInformation, _Out_ LPWSTR *StringSecurityDescriptor, _Out_ PULONG StringSecurityDescriptorLength) |
| titi    | BOOL | ConvertSidToStringSid(_In_ PSID Sid, _Out_ *StringSid)                                                                                                                                                                                                       |
| titi    | BOOL | ConvertSidToStringSidA(_In_ PSID Sid, _Out_ *StringSid)                                                                                                                                                                                                      |
| titi    | BOOL | ConvertSidToStringSidW(_In_ PSID Sid, _Out_ *StringSid)                                                                                                                                                                                                      |
| suiti   | BOOL | ConvertStringSecurityDescriptorToSecurityDescriptor(LPCTSTR StringSecurityDescriptor, _In_ StringSDRevision, _Out_ PSECURITY_DESCRIPTOR *SecurityDescriptor, _Out_ PULONG SecurityDescriptorLength)                                                          |
| auiti   | BOOL | ConvertStringSecurityDescriptorToSecurityDescriptorA(LPCSTR StringSecurityDescriptor, _In_ StringSDRevision, _Out_ PSECURITY_DESCRIPTOR *SecurityDescriptor, _Out_ PULONG SecurityDescriptorLength)                                                          |
| wuiti   | BOOL | ConvertStringSecurityDescriptorToSecurityDescriptorW(LPCWSTR StringSecurityDescriptor, _In_ StringSDRevision, _Out_ PSECURITY_DESCRIPTOR *SecurityDescriptor, _Out_ PULONG SecurityDescriptorLength)                                                         |
| stti    | BOOL | ConvertStringSidToSid(_In_ LPCTSTR StringSid, _Out_ *Sid)                                                                                                                                                                                                    |
| atiti   | BOOL | ConvertStringSidToSidA(_In_ LPCSTR StringSid, _Out_ *Sid)                                                                                                                                                                                                    |
| wtiti   | BOOL | ConvertStringSidToSidW(_In_ LPCWSTR StringSid, _Out_ *Sid)                                                                                                                                                                                                   |
| tttucti | BOOL | ConvertToAutolabelPrivateObjectSecurity(PSECURITY_DESCRIPTOR ParentDescriptor, PSECURITY_DESCRIPTOR CurrentDescriptor, PSECURITY_DESCRIPTOR *NewDescriptor)                                                                                                  |

|              |      |                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |      | GUID *ObjectType, _In_ BOOLEAN IsPGENERIC_MAPPING GenericMapping)                                                                                                                                                                                                                                                                                                                                      |
| uitti        | BOOL | CopySid(_In_ DWORD nDestinationSid, pDestinationSid, _In_ PSID pSourceSid)                                                                                                                                                                                                                                                                                                                             |
| tttitti      | BOOL | CreatePrivateObjectSecurity(_In_opt_ PSECURITY_DESCRIPTOR ParentDescriptor, PSECURITY_DESCRIPTOR CreatorDescriptor, PSECURITY_DESCRIPTOR *NewDescriptor, IsDirectoryObject, _In_opt_ HANDLE Token, PGENERIC_MAPPING GenericMapping)                                                                                                                                                                    |
| tttuiuitti   | BOOL | CreatePrivateObjectSecurityEx(_In_opt_ PSECURITY_DESCRIPTOR ParentDescriptor, PSECURITY_DESCRIPTOR CreatorDescriptor, PSECURITY_DESCRIPTOR *NewDescriptor, *ObjectType, _In_ BOOL IsContainerObject, AutoInheritFlags, _In_opt_ HANDLE Token, PGENERIC_MAPPING GenericMapping)                                                                                                                         |
| tttuiuitti   | BOOL | CreatePrivateObjectSecurityWithMultipleObjectTypes(_In_opt_ PSECURITY_DESCRIPTOR ParentDescriptor, PSECURITY_DESCRIPTOR CreatorDescriptor, PSECURITY_DESCRIPTOR *NewDescriptor, **ObjectTypes, _In_ ULONG GuidCount, IsContainerObject, _In_ ULONG AutoInheritFlags, _In_opt_ HANDLE Token, _In_ PGENERIC_MAPPING GenericMapping)                                                                      |
| tssttuitstti | BOOL | CreateProcessAsUser(_In_opt_ HANDLE hToken, LPCTSTR lpApplicationName, _Inout_opt_ LPCTSTR lpCommandLine, _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, _In_ BOOL bInheritHandles, dwCreationFlags, _In_opt_ LPVOID lpEnvironment, LPCTSTR lpCurrentDirectory, _In_ LPCTSTR lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation)  |
| taattuitatti | BOOL | CreateProcessAsUserA(_In_opt_ HANDLE hToken, LPCSTR lpApplicationName, _Inout_opt_ LPCTSTR lpCommandLine, _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, _In_ BOOL bInheritHandles, dwCreationFlags, _In_opt_ LPVOID lpEnvironment, LPCSTR lpCurrentDirectory, _In_ LPCTSTR lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation)   |
| twwtuitwtti  | BOOL | CreateProcessAsUserW(_In_opt_ HANDLE hToken, LPCWSTR lpApplicationName, _Inout_opt_ LPCTSTR lpCommandLine, _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, _In_ BOOL bInheritHandles, dwCreationFlags, _In_opt_ LPVOID lpEnvironment, LPCWSTR lpCurrentDirectory, _In_ LPCTSTR lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation) |

|                 |           |                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |           | LPCWSTR lpCurrentDirectory, _In_ LP lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation)                                                                                                                                                                                                                                      |
| wwwuiwwuitwti   | BOOL      | CreateProcessWithLogonW(_In_ LPCWSTR lpDomain, _In_ LPCTSTR lpApplicationName, _Inout_opt_ LPWSTR lpEnvironment, _In_opt_ LPCWSTR lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcess                                                                                                                                                    |
| tuiwwuitwti     | BOOL      | CreateProcessWithTokenW(_In_ HANDLE hToken, _In_ DWORD dwLogonFlags, _In_opt_ LPCWSTR lpApplicationName, _Inout_opt_ LPWSTR lpEnvironment, _In_opt_ LPCWSTR lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcess                                                                                                                          |
| tuiuituiti      | BOOL      | CreateRestrictedToken(_In_ HANDLE hToken, _In_ DWORD Flags, _In_ DWORD DiscretionarySidsToDisable, _In_ DWORD PrivilegesToDelete, _In_ DWORD RestrictSidsToRestrict, _Out_ HANDLE* NewTokenHandle)                                                                                                                                        |
| tssuiuiuisstst  | SC_HANDLE | CreateService(_In_ SC_HANDLE hSCManager, _In_opt_ LPCTSTR lpServiceName, _In_opt_ LPCTSTR lpDisplayName, _In_ DWORD dwDesiredAccess, _In_ DWORD dwStartType, _In_ LPCTSTR lpBinaryPathName, _In_opt_ LPCTSTR lpLoadOrderGroup, _Out_opt_ LPCTSTR lpDependencies, _In_opt_ LPCTSTR lpServiceStartName, _In_opt_ LPCTSTR                    |
| taaiuiuiiaataat | SC_HANDLE | CreateServiceA(_In_ SC_HANDLE hSCManager, _In_opt_ LPCSTR lpServiceName, _In_opt_ LPCSTR lpDisplayName, _In_ DWORD dwDesiredAccess, _In_ DWORD dwStartType, _In_ LPCSTR lpBinaryPathName, _In_opt_ LPCSTR lpLoadOrderGroup, _Out_opt_ LPCSTR lpDependencies, _In_opt_ LPCSTR lpServiceStartName, _In_opt_ LPCSTR                          |
| twwuiuiuiwwtwwt | SC_HANDLE | CreateServiceW(_In_ SC_HANDLE hSCManager, _In_opt_ LPCWSTR lpServiceName, _In_opt_ LPCWSTR lpDisplayName, _In_ DWORD dwDesiredAccess, _In_ DWORD dwServiceType, _In_ DWORD dwStartType, _In_ LPCWSTR lpBinaryPathName, _In_opt_ LPCWSTR lpLoadOrderGroup, _Out_opt_ LPCWSTR lpDependencies, _In_opt_ LPCWSTR lpServiceStartName, _In_opt_ |

|         |       |                                                                                                             |
|---------|-------|-------------------------------------------------------------------------------------------------------------|
|         |       | lpPassword)                                                                                                 |
| ttui    | ULONG | CreateTraceInstanceId(_In_ HANDLE R<br>PEVENT_INSTANCE_INFO pInstInfo)                                      |
| uittti  | BOOL  | CreateWellKnownSid(_In_ WELL_KNO<br>WellKnownSidType, _In_opt_ PSID Dor<br>PSID pSid, _Inout_ DWORD *cbSid) |
| suiiii  | BOOL  | CredDelete(_In_ LPCTSTR TargetName<br>_In_ DWORD Flags)                                                     |
| auuiii  | BOOL  | CredDeleteA(_In_ LPCSTR TargetName<br>_In_ DWORD Flags)                                                     |
| wuiiii  | BOOL  | CredDeleteW(_In_ LPCWSTR TargetNa<br>Type, _In_ DWORD Flags)                                                |
| suiitti | BOOL  | CredEnumerate(_In_ LPCTSTR Filter, _<br>_Out_ DWORD *Count, _Out_ PCREDI                                    |
| auitti  | BOOL  | CredEnumerateA(_In_ LPCSTR Filter, _<br>_Out_ DWORD *Count, _Out_ PCREDI                                    |
| wuitti  | BOOL  | CredEnumerateW(_In_ LPCWSTR Filte<br>_Out_ DWORD *Count, _Out_ PCREDI                                       |
| suiiiti | BOOL  | CredFindBestCredential(_In_ LPCTSTR<br>DWORD Type, _In_ DWORD Flags, _C<br>*Credential)                     |
| auiiti  | BOOL  | CredFindBestCredentialA(_In_ LPCSTR<br>DWORD Type, _In_ DWORD Flags, _C<br>*Credential)                     |
| wuiiiti | BOOL  | CredFindBestCredentialW(_In_ LPCWS<br>DWORD Type, _In_ DWORD Flags, _C<br>*Credential)                      |
| ti      | VOID  | CredFree(_In_ PVOID Buffer)                                                                                 |
| uiti    | BOOL  | CredGetSessionTypes(_In_ DWORD Ma<br>_Out_ LPDWORD MaximumPersist)                                          |
| suiti   | BOOL  | CredGetTargetInfo(_In_ LPCTSTR Targ<br>Flags, _Out_ PCREDENTIAL_TARGET<br>*TargetInfo)                      |
| auiti   | BOOL  | CredGetTargetInfoA(_In_ LPCSTR Targ<br>Flags, _Out_ PCREDENTIAL_TARGET<br>*TargetInfo)                      |
| wuiti   | BOOL  | CredGetTargetInfoW(_In_ LPCWSTR T<br>DWORD Flags, _Out_<br>PCREDENTIAL_TARGET_INFORMA                       |
| si      | BOOL  | CredIsMarshaledCredential(_In_ LPCTS<br>MarshaledCredential)                                                |
| ai      | BOOL  | CredIsMarshaledCredentialA(_In_ LPCS<br>MarshaledCredential)                                                |

|          |      |                                                                                                                                                        |
|----------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| wi       | BOOL | CredIsMarshaledCredentialW(_In_ LPC<br>MarshaledCredential)                                                                                            |
| sti      | BOOL | CredIsProtected(_In_ LPTSTR pszProtec<br>CRED_PROTECTION_TYPE *pProtec                                                                                 |
| ati      | BOOL | CredIsProtectedA(_In_ LPSTR pszProte<br>CRED_PROTECTION_TYPE *pProtec                                                                                  |
| wti      | BOOL | CredIsProtectedW(_In_ LPWSTR pszPr<br>_Out_ CRED_PROTECTION_TYPE *p                                                                                    |
| uitti    | BOOL | CredMarshalCredential(_In_ CRED_MA<br>CredType, _In_ PVOID Credential, _Out<br>*MarshaledCredential)                                                   |
| uitti    | BOOL | CredMarshalCredentialA(_In_ CRED_M<br>CredType, _In_ PVOID Credential, _Out<br>*MarshaledCredential)                                                   |
| uitti    | BOOL | CredMarshalCredentialW(_In_ CRED_M<br>CredType, _In_ PVOID Credential, _Out<br>*MarshaledCredential)                                                   |
| tti      | int  | CredMarshalTargetInfo(_In_<br>PCREDENTIAL_TARGET_INFORMA<br>_Out_ PUSHORT *Buffer, PULONG Bu                                                           |
| isuisti  | BOOL | CredProtect(_In_ BOOL fAsSelf, _In_ L<br>_In_ DWORD cchCredentials, _Out_ LP<br>pszProtectedCredentials, _Inout_ DWOF<br>_Out_ CRED_PROTECTION_TYPE *P |
| iauiatti | BOOL | CredProtectA(_In_ BOOL fAsSelf, _In_<br>_In_ DWORD cchCredentials, _Out_ LP<br>pszProtectedCredentials, _Inout_ DWOF<br>_Out_ CRED_PROTECTION_TYPE *P  |
| iwuiwtti | BOOL | CredProtectW(_In_ BOOL fAsSelf, _In_<br>pszCredentials, _In_ DWORD cchCrede<br>pszProtectedCredentials, _Inout_ DWOF<br>_Out_ CRED_PROTECTION_TYPE *P  |
| suiuiti  | BOOL | CredRead(_In_ LPCTSTR TargetName,<br>_In_ DWORD Flags, _Out_ PCREDENT                                                                                  |
| auiuiti  | BOOL | CredReadA(_In_ LPCSTR TargetName,<br>_In_ DWORD Flags, _Out_ PCREDENT                                                                                  |
| tuitti   | BOOL | CredReadDomainCredentials(_In_<br>PCREDENTIAL_TARGET_INFORMA<br>DWORD Flags, _Out_ DWORD *Count<br>**Credentials)                                      |
| tuitti   | BOOL | CredReadDomainCredentialsA(_In_<br>PCREDENTIAL_TARGET_INFORMA<br>DWORD Flags, _Out_ DWORD *Count<br>**Credentials)                                     |
|          |      |                                                                                                                                                        |

|         |      |                                                                                                                                                |
|---------|------|------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti  | BOOL | CredReadDomainCredentialsW(_In_ PCREDENTIAL_TARGET_INFORMATION Flags, _Out_ DWORD *Count **Credentials)                                        |
| wuiuiti | BOOL | CredReadW(_In_ LPCWSTR TargetName, _In_ DWORD Flags, _Out_ PCREDENTIAL)                                                                        |
| ssuiiii | BOOL | CredRename(_In_ LPCTSTR OldTargetName, _In_ LPCWSTR NewTargetName, _In_ DWORD Type, _Out_ BOOL *Success)                                       |
| aauiiii | BOOL | CredRenameA(_In_ LPCSTR OldTargetName, _In_ LPCWSTR NewTargetName, _In_ DWORD Type, _Out_ BOOL *Success)                                       |
| wwuiiii | BOOL | CredRenameW(_In_ LPCWSTR OldTargetName, _In_ LPCWSTR NewTargetName, _In_ DWORD Type, _Out_ BOOL *Success)                                      |
| stti    | BOOL | CredUnmarshalCredential(_In_ LPCTSTR TargetName, _Out_ PCRED_MARSHAL_TYPE Credential, _Out_ BOOL *Success)                                     |
| atti    | BOOL | CredUnmarshalCredentialA(_In_ LPCSTR TargetName, _Out_ PCRED_MARSHAL_TYPE Credential, _Out_ BOOL *Success)                                     |
| wtti    | BOOL | CredUnmarshalCredentialW(_In_ LPCWSTR TargetName, _Out_ PCRED_MARSHAL_TYPE Credential, _Out_ BOOL *Success)                                    |
| isuisti | BOOL | CredUnprotect(_In_ BOOL fAsSelf, _In_ LPCTSTR pszProtectedCredentials, _In_ DWORD Flags, _Out_ LPCTSTR pszCredentials, _Inout_ DWORD *Length)  |
| iauiati | BOOL | CredUnprotectA(_In_ BOOL fAsSelf, _In_ LPCTSTR pszProtectedCredentials, _In_ DWORD Flags, _Out_ LPCTSTR pszCredentials, _Inout_ DWORD *Length) |
| iwuiwti | BOOL | CredUnprotectW(_In_ BOOL fAsSelf, _In_ LPWSTR pszProtectedCredentials, _In_ DWORD Flags, _Out_ LPWSTR pszCredentials, _Inout_ DWORD *Length)   |
| tuii    | BOOL | CredWrite(_In_ PCREDENTIAL Credential, _In_ DWORD Flags)                                                                                       |
| tuii    | BOOL | CredWriteA(_In_ PCREDENTIAL Credential, _In_ DWORD Flags)                                                                                      |
| ttuii   | BOOL | CredWriteDomainCredentials(_In_ PCREDENTIAL_TARGET_INFORMATION Flags, _In_ PCREDENTIAL Credential, _In_ DWORD Count, _Out_ DWORD *Count)       |
| ttuii   | BOOL | CredWriteDomainCredentialsA(_In_ PCREDENTIAL_TARGET_INFORMATION Flags, _In_ PCREDENTIAL Credential, _In_ DWORD Count, _Out_ DWORD *Count)      |
| ttuii   | BOOL | CredWriteDomainCredentialsW(_In_ PCREDENTIAL_TARGET_INFORMATION Flags, _In_ PCREDENTIAL Credential, _In_ DWORD Count, _Out_ DWORD *Count)      |

|            |      |                                                                                                                              |
|------------|------|------------------------------------------------------------------------------------------------------------------------------|
| tuii       | BOOL | CredWriteW(_In_ PCREDENTIAL Cred<br>Flags)                                                                                   |
| tssuiiii   | BOOL | CryptAcquireContext(_Out_ HCRYPTP<br>LPCTSTR pszContainer, _In_ LPCTSTR<br>DWORD dwProvType, _In_ DWORD d                    |
| taaiuiii   | BOOL | CryptAcquireContextA(_Out_ HCRYPT<br>LPCSTR pszContainer, _In_ LPCSTR ps<br>DWORD dwProvType, _In_ DWORD d                   |
| twwuiiii   | BOOL | CryptAcquireContextW(_Out_ HCRYPT<br>LPCWSTR pszContainer, _In_ LPCWST<br>DWORD dwProvType, _In_ DWORD d                     |
| ttuii      | BOOL | CryptContextAddRef(_In_ HCRYPTPRO<br>DWORD *pdwReserved, _In_ DWORD                                                          |
| tuituiti   | BOOL | CryptCreateHash(_In_ HCRYPTPROV I<br>Algid, _In_ HCRYPTKEY hKey, _In_ D<br>HCRYPTHASH *phHash)                               |
| ttiuitti   | BOOL | CryptDecrypt(_In_ HCRYPTKEY hKey,<br>hHash, _In_ BOOL Final, _In_ DWORD<br>BYTE *pbData, _Inout_ DWORD *pdw                  |
| tuituiti   | BOOL | CryptDeriveKey(_In_ HCRYPTPROV h<br>Algid, _In_ HCRYPTHASH hBaseData,<br>_Inout_ HCRYPTKEY *phKey)                           |
| ti         | BOOL | CryptDestroyHash(_In_ HCRYPTHASH                                                                                             |
| ti         | BOOL | CryptDestroyKey(_In_ HCRYPTKEY h                                                                                             |
| ttuiti     | BOOL | CryptDuplicateHash(_In_ HCRYPTHAS<br>*pdwReserved, _In_ DWORD dwFlags,<br>*phHash)                                           |
| ttuiti     | BOOL | CryptDuplicateKey(_In_ HCRYPTKEY<br>*pdwReserved, _In_ DWORD dwFlags,<br>*phKey)                                             |
| ttiuittuii | BOOL | CryptEncrypt(_In_ HCRYPTKEY hKey,<br>hHash, _In_ BOOL Final, _In_ DWORD<br>BYTE *pbData, _Inout_ DWORD *pdw<br>dwBufLen)     |
| uituitsti  | BOOL | CryptEnumProviders(_In_ DWORD dw<br>*pdwReserved, _In_ DWORD dwFlags,<br>*pdwProvType, _Out_ LPTSTR pszProv<br>*pcbProvName) |
| uituitati  | BOOL | CryptEnumProvidersA(_In_ DWORD d<br>*pdwReserved, _In_ DWORD dwFlags,<br>*pdwProvType, _Out_ LPSTR pszProvN<br>*pcbProvName) |
| uituitwti  | BOOL | CryptEnumProvidersW(_In_ DWORD d<br>*pdwReserved, _In_ DWORD dwFlags,<br>*pdwProvType, _Out_ LPWSTR pszPro                   |

|           |      |                                                                                                                                  |
|-----------|------|----------------------------------------------------------------------------------------------------------------------------------|
|           |      | DWORD *pcbProvName)                                                                                                              |
| uituitsti | BOOL | CryptEnumProviderTypes(_In_ DWORD *pdwReserved, _In_ DWORD *pdwProvType, _Out_ LPTSTR *pcbProvName, _Inout_ DWORD *pcbProvName)  |
| uituitati | BOOL | CryptEnumProviderTypesA(_In_ DWORD *pdwReserved, _In_ DWORD *pdwProvType, _Out_ LPSTR *pcbProvName, _Inout_ DWORD *pcbProvName)  |
| uituitwti | BOOL | CryptEnumProviderTypesW(_In_ DWORD *pdwReserved, _In_ DWORD *pdwProvType, _Out_ LPWSTR *pcbProvName, _Inout_ DWORD *pcbProvName) |
| ttuiuiti  | BOOL | CryptExportKey(_In_ HCRYPTKEY hKey, _In_ DWORD dwBlobType, _Out_ BYTE *pbData, _Inout_ DWORD *pcbData)                           |
| tuiuiti   | BOOL | CryptGenKey(_In_ HCRYPTPROV hProv, _In_ DWORD dwFlags, _Out_ HCRYPTKEY *phKey)                                                   |
| tuiti     | BOOL | CryptGenRandom(_In_ HCRYPTPROV hProv, dwLen, _Inout_ BYTE *pbBuffer)                                                             |
| uituisti  | BOOL | CryptGetDefaultProvider(_In_ DWORD *pdwReserved, _In_ DWORD *pdwProvType, _Out_ LPTSTR pszProvName, _Inout_ DWORD *pcbProvName)  |
| uituiati  | BOOL | CryptGetDefaultProviderA(_In_ DWORD *pdwReserved, _In_ DWORD *pdwProvType, _Out_ LPSTR pszProvName, _Inout_ DWORD *pcbProvName)  |
| uituiwti  | BOOL | CryptGetDefaultProviderW(_In_ DWORD *pdwReserved, _In_ DWORD *pdwProvType, _Out_ LPWSTR pszProvName, _Inout_ DWORD *pcbProvName) |
| tuittuii  | BOOL | CryptGetHashParam(_In_ HCRYPTHASH hHash, _In_ DWORD dwParam, _Out_ BYTE *pbData, _Inout_ DWORD *pdwDataLen, _In_ DWORD dwFlags)  |
| tuittuii  | BOOL | CryptGetKeyParam(_In_ HCRYPTKEY hKey, _In_ DWORD dwParam, _Out_ BYTE *pbData, _Inout_ DWORD *pdwDataLen, _In_ DWORD dwFlags)     |
| tuittuii  | BOOL | CryptGetProvParam(_In_ HCRYPTPROV hProv, _In_ DWORD dwParam, _Out_ BYTE *pbData, _Inout_ DWORD *pdwDataLen, _In_ DWORD dwFlags)  |
| tuiti     | BOOL | CryptGetUserKey(_In_ HCRYPTPROV hProv, dwKeySpec, _Out_ HCRYPTKEY *phKey)                                                        |
| ttuiuii   | BOOL | CryptHashData(_In_ HCRYPTHASH hHash, _In_ BYTE *pbData, _In_ DWORD dwDataLen, _In_ DWORD dwFlags)                                |
| ttuii     | BOOL | CryptHashSessionKey(_In_ HCRYPTHASH hHash, _In_ HCRYPTKEY hKey, _In_ DWORD dwDataLen, _Out_ BYTE *pbData)                        |

|            |      |                                                                                                                                                                                                                |
|------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuituiti  | BOOL | CryptImportKey(_In_ HCRYPTPROV hProv, _In_ const BYTE *pbData, _In_ DWORD dwDataLen, _In_ HCRYPTKEY hPubKey, _In_ DWORD dwFlags, _Out_ HCRYPTKEY *phKey)                                                       |
| tuii       | BOOL | CryptReleaseContext(_In_ HCRYPTPROV hProv, _In_ DWORD dwFlags)                                                                                                                                                 |
| tuituii    | BOOL | CryptSetHashParam(_In_ HCRYPTHASH hCryptHash, _In_ DWORD dwParam, _In_ const BYTE *pbData, _In_ DWORD dwFlags)                                                                                                 |
| tuituii    | BOOL | CryptSetKeyParam(_In_ HCRYPTKEY hCryptKey, _In_ DWORD dwParam, _In_ const BYTE *pbData, _In_ DWORD dwFlags)                                                                                                    |
| suii       | BOOL | CryptSetProvider(_In_ LPCTSTR pszProvName, _In_ DWORD dwProvType)                                                                                                                                              |
| auii       | BOOL | CryptSetProviderA(_In_ LPCSTR pszProvName, _In_ DWORD dwProvType)                                                                                                                                              |
| suituii    | BOOL | CryptSetProviderEx(_In_ LPCTSTR pszProvName, _In_ DWORD dwProvType, _In_ DWORD dwFlags)                                                                                                                        |
| auituii    | BOOL | CryptSetProviderExA(_In_ LPCSTR pszProvName, _In_ DWORD dwProvType, _In_ DWORD dwFlags)                                                                                                                        |
| wuituii    | BOOL | CryptSetProviderExW(_In_ LPCWSTR pszProvName, _In_ DWORD dwProvType, _In_ DWORD dwFlags)                                                                                                                       |
| wuii       | BOOL | CryptSetProviderW(_In_ LPCWSTR pszProvName, _In_ DWORD dwProvType)                                                                                                                                             |
| tuituii    | BOOL | CryptSetProvParam(_In_ HCRYPTPROV hCryptProv, _In_ DWORD dwParam, _In_ const BYTE *pbData, _In_ DWORD dwFlags)                                                                                                 |
| tuisuitti  | BOOL | CryptSignHash(_In_ HCRYPTHASH hCryptHash, _In_ const BYTE *pbData, _In_ DWORD dwKeySpec, _In_ LPCTSTR sDescription, _In_ DWORD dwFlags, _Out_ BYTE *pbSignature, _In_ DWORD dwSigLen, _Out_ DWORD *pdwSigLen)  |
| tuiasuitti | BOOL | CryptSignHashA(_In_ HCRYPTHASH hCryptHash, _In_ const BYTE *pbData, _In_ LPCSTR sDescription, _In_ DWORD dwKeySpec, _In_ DWORD dwFlags, _Out_ BYTE *pbSignature, _In_ DWORD dwSigLen, _Out_ DWORD *pdwSigLen)  |
| tuiwuiitti | BOOL | CryptSignHashW(_In_ HCRYPTHASH hCryptHash, _In_ const BYTE *pbData, _In_ LPCWSTR sDescription, _In_ DWORD dwKeySpec, _In_ DWORD dwFlags, _Out_ BYTE *pbSignature, _In_ DWORD dwSigLen, _Out_ DWORD *pdwSigLen) |
| ttuitsuii  | BOOL | CryptVerifySignature(_In_ HCRYPTKEY hCryptKey, _In_ const BYTE *pbSignature, _In_ DWORD dwSigLen, _In_ HCRYPTPROV hCryptProv, _In_ LPCTSTR sDescription)                                                       |
| ttuitauii  | BOOL | CryptVerifySignatureA(_In_ HCRYPTKEY hCryptKey, _In_ const BYTE *pbSignature, _In_ DWORD dwSigLen, _In_ HCRYPTPROV hCryptProv, _In_ LPCSTR sDescription)                                                       |

|                     |       |                                                                                                                                                                                                                            |
|---------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuitwuii           | BOOL  | CryptVerifySignatureW(_In_ HCRYPTO<br>BYTE *pbSignature, _In_ DWORD dwS<br>HCRYPTKEY hPubKey, _In_ LPCWSTR<br>DWORD dwFlags)                                                                                               |
| suii                | BOOL  | DecryptFile(_In_ LPCTSTR lpFileName<br>dwReserved)                                                                                                                                                                         |
| auii                | BOOL  | DecryptFileA(_In_ LPCSTR lpFileName<br>dwReserved)                                                                                                                                                                         |
| wuii                | BOOL  | DecryptFileW(_In_ LPCWSTR lpFileNa<br>DWORD dwReserved)                                                                                                                                                                    |
| tuii                | BOOL  | DeleteAce(_Inout_ PACL pAcl, _In_ DV                                                                                                                                                                                       |
| ti                  | BOOL  | DeleteService(_In_ SC_HANDLE hServ                                                                                                                                                                                         |
| ti                  | BOOL  | DeregisterEventSource(_Inout_ HANDL                                                                                                                                                                                        |
| ti                  | BOOL  | DestroyPrivateObjectSecurity(_Inout_<br>PSECURITY_DESCRIPTOR *ObjectD                                                                                                                                                      |
| ssuiuitui           | DWORD | DuplicateEncryptionInfoFile(_In_ LPCT<br>LPCTSTR DstFileName, _In_ DWORD<br>_In_ DWORD dwAttributes, _In_opt_ c<br>LPSECURITY_ATTRIBUTES lpSecurit                                                                         |
| tuiti               | BOOL  | DuplicateToken(_In_ HANDLE Existing<br>SECURITY_IMPERSONATION_LEVE<br>_Out_ PHANDLE DuplicateTokenHand                                                                                                                     |
| tuituiuiti          | BOOL  | DuplicateTokenEx(_In_ HANDLE hExis<br>DWORD dwDesiredAccess, _In_opt_<br>LPSECURITY_ATTRIBUTES lpToken/<br>SECURITY_IMPERSONATION_LEVE<br>_In_ TOKEN_TYPE TokenType, _Out_<br>phNewToken)                                  |
| uiuiuittui          | ULONG | EnableTrace(_In_ ULONG Enable, _In_<br>_In_ ULONG EnableLevel, _In_ LPCGU<br>TRACEHANDLE SessionHandle)                                                                                                                    |
| tttuiuicui6ui6uitui | ULONG | EnableTraceEx(_In_ LPCGUID Provide<br>SourceId, _In_ TRACEHANDLE TraceL<br>IsEnabled, _In_ UCHAR Level, _In_ UI<br>MatchAnyKeyword, _In_ ULONGLONG<br>_In_ ULONG EnableProperty, _In_opt_<br>PEVENT_FILTER_DESCRIPTOR Enab |
| si                  | BOOL  | EncryptFile(_In_ LPCTSTR lpFileName                                                                                                                                                                                        |
| ai                  | BOOL  | EncryptFileA(_In_ LPCSTR lpFileName                                                                                                                                                                                        |
| wi                  | BOOL  | EncryptFileW(_In_ LPCWSTR lpFileNa                                                                                                                                                                                         |
| wii                 | BOOL  | EncryptionDisable(_In_ LPCWSTR DirI<br>Disable)                                                                                                                                                                            |
|                     |       | EnumDependentServices(_In_ SC_HAN                                                                                                                                                                                          |

|               |       |                                                                                                                                                                                                                |
|---------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuituitti     | BOOL  | DWORD dwServiceState, _Out_opt_ LPENUM_SERVICE_STATUS lpService, cbBufSize, _Out_ LPDWORD pcbBytes, LPDWORD lpServicesReturned)                                                                                |
| tuituitti     | BOOL  | EnumDependentServicesA(_In_ SC_HANDLE hService, _Out_opt_ DWORD dwServiceState, _Out_opt_ LPENUM_SERVICE_STATUS lpService, cbBufSize, _Out_ LPDWORD pcbBytes, LPDWORD lpServicesReturned)                      |
| tuituitti     | BOOL  | EnumDependentServicesW(_In_ SC_HANDLE hService, _Out_opt_ DWORD dwServiceState, _Out_opt_ LPENUM_SERVICE_STATUS lpService, cbBufSize, _Out_ LPDWORD pcbBytes, LPDWORD lpServicesReturned)                      |
| tuitui        | ULONG | EnumerateTraceGuids(_Inout_ PTRACEGUID *GuidPropertiesArray, _In_ ULONG ProcId, PULONG GuidCount)                                                                                                              |
| uituituitui   | ULONG | EnumerateTraceGuidsEx(_In_ TRACE_QUERY_INFO_CLASS TraceQueryInfoClass, _In_ PVOID InBuffer, InBufferSize, _Out_ PVOID OutBuffer, OutBufferSize, _Out_ PULONG ReturnCount)                                      |
| tuiuituitti   | BOOL  | EnumServicesStatus(_In_ SC_HANDLE hService, _Out_opt_ LPENUM_SERVICE_STATUS lpServicesReturned, _Inout_ lpResumeHandle)                                                                                        |
| tuiuituitti   | BOOL  | EnumServicesStatusA(_In_ SC_HANDLE hService, _Out_opt_ LPENUM_SERVICE_STATUS lpServicesReturned, _Inout_ lpResumeHandle)                                                                                       |
| tiuiuituittsi | BOOL  | EnumServicesStatusEx(_In_ SC_HANDLE hService, SC_ENUM_TYPE InfoLevel, _In_ DWORD dwServiceState, _Out_opt_ _In_ DWORD cbBufSize, _Out_ LPDWORD lpServicesReturned, lpResumeHandle, _In_opt_ LPCTSTR pszFilter) |
| tiuiuituittai | BOOL  | EnumServicesStatusExA(_In_ SC_HANDLE hService, SC_ENUM_TYPE InfoLevel, _In_ DWORD dwServiceState, _Out_opt_ _In_ DWORD cbBufSize, _Out_ LPDWORD lpServicesReturned, lpResumeHandle, _In_opt_ LPCSTR pszFilter) |
|               |       | EnumServicesStatusExW(_In_ SC_HANDLE hService, SC_ENUM_TYPE InfoLevel, _In_ DWORD dwServiceState, _Out_opt_                                                                                                    |

|                 |         |                                                                                                                                                                                                                                                |
|-----------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiuiuitittwi    | BOOL    | <a href="#">_In_ DWORD cbBufSize, _Out_ LPDWORD lpServicesReturned, _Out_ LPDWORD lpResumeHandle, _In_opt_ LPCWSTR lpServiceName</a>                                                                                                           |
| tuiuititti      | BOOL    | <a href="#">EnumServicesStatusW(_In_ SC_HANDLE hService, _In_ DWORD dwServiceType, _In_ DWORD dwLevel, _Out_opt_ LPENUM_SERVICE_STATUS lpServicesReturned, _Inout_ LPDWORD lpServicesReturned, _Inout_ LPDWORD lpResumeHandle)</a>             |
| titi            | BOOL    | <a href="#">EqualDomainSid(_In_ PSID pSid1, _In_ PSID pSid2, _Out_ BOOL *pfEqual)</a>                                                                                                                                                          |
| titi            | BOOL    | <a href="#">EqualPrefixSid(_In_ PSID pSid1, _In_ PSID pSid2, _Out_ BOOL *pfEqual)</a>                                                                                                                                                          |
| titi            | BOOL    | <a href="#">EqualSid(_In_ PSID pSid1, _In_ PSID pSid2, _Out_ BOOL *pfEqual)</a>                                                                                                                                                                |
| tttui           | ULONG   | <a href="#">EventAccessQuery(_In_ LPGUID Guid, _In_ PSECURITY_DESCRIPTOR Buffer, _Out_ ULONG *pAccess, _In_ ULONG BufferSize)</a>                                                                                                              |
| uitui           | ULONG   | <a href="#">EventActivityIdControl(_In_ ULONG Control, _In_ LPGUID ActivityId)</a>                                                                                                                                                             |
| i6tuc           | BOOLEAN | <a href="#">EventEnabled(_In_ REGHANDLE RegHandle, _In_ PCEVENT_DESCRIPTOR EventDescriptor, _Out_ BOOLEAN *pEnabled)</a>                                                                                                                       |
| i6ucui6uc       | BOOLEAN | <a href="#">EventProviderEnabled(_In_ REGHANDLE RegHandle, _In_ UCHAR Level, _In_ ULONGLONG Key, _Out_ BOOLEAN *pEnabled)</a>                                                                                                                  |
| tttui           | ULONG   | <a href="#">EventRegister(_In_ LPCGUID ProviderId, _In_ PENABLECALLBACK EnableCallback, _In_ CallbackContext, _Out_ PREGHANDLE *ppRegHandle)</a>                                                                                               |
| i6ui            | ULONG   | <a href="#">EventUnregister(_In_ REGHANDLE RegHandle, _Out_ ULONG *pProviderId)</a>                                                                                                                                                            |
| i6tuitui        | ULONG   | <a href="#">EventWrite(_In_ REGHANDLE RegHandle, _In_ PCEVENT_DESCRIPTOR EventDescriptor, _In_ ULONG UserDataCount, _In_opt_ PEVENT_DATA_DESCRIPTOR *pUserData)</a>                                                                            |
| i6tui6uittuitui | ULONG   | <a href="#">EventWriteEx(_In_ REGHANDLE RegHandle, _In_ PCEVENT_DESCRIPTOR EventDescriptor, _In_ ULONG Filter, _In_ ULONG Flags, _In_opt_ LPCGUID RelatedActivityId, _In_ ULONG UserDataCount, _In_opt_ PEVENT_DATA_DESCRIPTOR *pUserData)</a> |
| i6ucui6wui      | ULONG   | <a href="#">EventWriteString(_In_ REGHANDLE RegHandle, _In_ UCHAR Level, _In_ ULONGLONG Keyword, _In_ LPCTSTR lpString, _Out_ ULONG *pLength)</a>                                                                                              |
| i6tttuitui      | ULONG   | <a href="#">EventWriteTransfer(_In_ REGHANDLE RegHandle, _In_ PCEVENT_DESCRIPTOR EventDescriptor, _In_ LPCGUID ActivityId, _In_ LPGUID RelatedActivityId, _In_ ULONG UserDataCount, _In_opt_ PEVENT_DATA_DESCRIPTOR *pUserData)</a>            |
| sti             | BOOL    | <a href="#">FileEncryptionStatus(_In_ LPCTSTR lpFileName, _Out_ LPDWORD lpStatus)</a>                                                                                                                                                          |

|        |       |                                                                                                                               |
|--------|-------|-------------------------------------------------------------------------------------------------------------------------------|
| ati    | BOOL  | FileEncryptionStatusA(_In_ LPCSTR lpLPDWORD lpStatus)                                                                         |
| wti    | BOOL  | FileEncryptionStatusW(_In_ LPCWSTR LPDWORD lpStatus)                                                                          |
| tti    | BOOL  | FindFirstFreeAce(_In_ PACL pAcl, _Out_                                                                                        |
| tstui  | ULONG | FlushTrace(_In_ TRACEHANDLE SessLPCTSTR SessionName, _Inout_ PEVENT_TRACE_PROPERTIES Prop                                     |
| tatui  | ULONG | FlushTraceA(_In_ TRACEHANDLE SeLPCSTR SessionName, _Inout_ PEVENT_TRACE_PROPERTIES Prop                                       |
| twtui  | ULONG | FlushTraceW(_In_ TRACEHANDLE SeLPCWSTR SessionName, _Inout_ PEVENT_TRACE_PROPERTIES Prop                                      |
| ti     | VOID  | FreeEncryptionCertificateHashList(_In_ PENCRYPTION_CERTIFICATE_HASH                                                           |
| tuhtui | DWORD | FreeInheritedFromArray(_In_ PINHERI pInheritArray, _In_ USHORT AceCnt, _ PFN_OBJECT_MGR_FUNCTS pfnArr                         |
| tt     | PVOID | FreeSid(_In_ PSID pSid)                                                                                                       |
| tuiti  | BOOL  | GetAce(_In_ PACL pAcl, _In_ DWORD LPVOID *pAce)                                                                               |
| ttuiii | BOOL  | GetAclInformation(_In_ PACL pAcl, _O pAclInformation, _In_ DWORD nAclInf ACL_INFORMATION_CLASS dwAclI                         |
| ttttui | DWORD | GetAuditedPermissionsFromAcl(_In_ P PTRUSTEE pTrustee, _Out_ PACCESS pSuccessfulAuditedRights, _Out_ PACC pFailedAuditRights) |
| ttttui | DWORD | GetAuditedPermissionsFromAclA(_In_ PTRUSTEE pTrustee, _Out_ PACCESS pSuccessfulAuditedRights, _Out_ PACC pFailedAuditRights)  |
| ttttui | DWORD | GetAuditedPermissionsFromAclW(_In_ PTRUSTEE pTrustee, _Out_ PACCESS pSuccessfulAuditedRights, _Out_ PACC pFailedAuditRights)  |
| ti     | BOOL  | GetCurrentHwProfile(_Out_ LPHW_PR lpHwProfileInfo)                                                                            |
| ti     | BOOL  | GetCurrentHwProfileA(_Out_ LPHW_P lpHwProfileInfo)                                                                            |
| ti     | BOOL  | GetCurrentHwProfileW(_Out_ LPHW_I lpHwProfileInfo)                                                                            |
|        |       |                                                                                                                               |

|                 |       |                                                                                                                                                                                                                                                                              |
|-----------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tttui           | DWORD | GetEffectiveRightsFromAcl(_In_ PACL pTrustee, _Out_ PACCESS_MASK pAc                                                                                                                                                                                                         |
| tttui           | DWORD | GetEffectiveRightsFromAclA(_In_ PACL pTrustee, _Out_ PACCESS_MASK pAc                                                                                                                                                                                                        |
| tttui           | DWORD | GetEffectiveRightsFromAclW(_In_ PACL PTRUSTEE pTrustee, _Out_ PACCESS                                                                                                                                                                                                        |
| tuituiti        | BOOL  | GetEventLogInformation(_In_ HANDLE DWORD dwInfoLevel, _Out_ LPVOID cbBufSize, _Out_ LPDWORD pcbBytes                                                                                                                                                                         |
| tttui           | DWORD | GetExplicitEntriesFromAcl(_In_ PACL ppcCountOfExplicitEntries, _Out_ PEXPI *pListOfExplicitEntries)                                                                                                                                                                          |
| tttui           | DWORD | GetExplicitEntriesFromAclA(_In_ PACL ppcCountOfExplicitEntries, _Out_ PEXPI *pListOfExplicitEntries)                                                                                                                                                                         |
| tttui           | DWORD | GetExplicitEntriesFromAclW(_In_ PACL ppcCountOfExplicitEntries, _Out_ PEXPI *pListOfExplicitEntries)                                                                                                                                                                         |
| suituiti        | BOOL  | GetFileSecurity(_In_ LPCTSTR lpFileName SECURITY_INFORMATION Requested PSECURITY_DESCRIPTOR pSecurityI DWORD nLength, _Out_ LPDWORD lp                                                                                                                                       |
| autuiti         | BOOL  | GetFileSecurityA(_In_ LPCSTR lpFileName SECURITY_INFORMATION Requested PSECURITY_DESCRIPTOR pSecurityI DWORD nLength, _Out_ LPDWORD lp                                                                                                                                       |
| wuituiti        | BOOL  | GetFileSecurityW(_In_ LPCWSTR lpFileName SECURITY_INFORMATION Requested PSECURITY_DESCRIPTOR pSecurityI DWORD nLength, _Out_ LPDWORD lp                                                                                                                                      |
| suiiuititttui   | DWORD | GetInheritanceSource(_In_ LPTSTR pObject SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityI Container, _In_opt_ GUID **pObjectCl GuidCount, _In_ PACL pAcl, _In_opt_ PFN_OBJECT_MGR_FUNCTS pfnArra PGENERIC_MAPPING pGenericMappi PINHERITED_FROM pInheritArray) |
| aiiuiiuititttui | DWORD | GetInheritanceSourceA(_In_ LPSTR pObject SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityI Container, _In_opt_ GUID **pObjectCl GuidCount, _In_ PACL pAcl, _In_opt_ PFN_OBJECT_MGR_FUNCTS pfnArra PGENERIC_MAPPING pGenericMappi PINHERITED_FROM pInheritArray) |
|                 |       |                                                                                                                                                                                                                                                                              |

|               |       |                                                                                                                                                                                                                                                                                        |
|---------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wuiiituitttui | DWORD | GetInheritanceSourceW(_In_ LPWSTR SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation Container, _In_opt_ GUID **pObjectClassGuidCount, _In_ PACL pAcl, _In_opt_ PFN_OBJECT_MGR_FUNCTS pfnArrangedPGENERIC_MAPPING pGenericMapping PINHERITED_FROM pInheritArray) |
| tuituiti      | BOOL  | GetKernelObjectSecurity(_In_ HANDLE SECURITY_INFORMATION RequestedPSECURITY_DESCRIPTOR pSecurityDescriptor DWORD nLength, _Out_ LPDWORD lp                                                                                                                                             |
| tui           | DWORD | GetLengthSid(_In_ PSID pSid)                                                                                                                                                                                                                                                           |
| ittui         | DWORD | GetLocalManagedApplications(_In_ BOOL LPDWORD pdwApps, _Out_ PLOCALMANAGEDAPPLICATION *p                                                                                                                                                                                               |
| uitui         | DWORD | GetManagedApplicationCategories(_Out_ _Out_ APPCATEGORYINFOLIST *pA                                                                                                                                                                                                                    |
| tuiiuttui     | DWORD | GetManagedApplications(_In_ GUID *p dwQueryFlags, _In_ DWORD dwInfoLevel pdwApps, _Out_ PMANAGEDAPPLIC *prgManagedApps)                                                                                                                                                                |
| suiiittttui   | DWORD | GetNamedSecurityInfo(_In_ LPTSTR p SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation *ppsidOwner, _Out_opt_ PSID *ppsidGroup *ppDacl, _Out_opt_ PACL *ppSacl, _Out PSECURITY_DESCRIPTOR *ppSecurity                                                             |
| aiiuittttui   | DWORD | GetNamedSecurityInfoA(_In_ LPSTR p SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation *ppsidOwner, _Out_opt_ PSID *ppsidGroup *ppDacl, _Out_opt_ PACL *ppSacl, _Out PSECURITY_DESCRIPTOR *ppSecurity                                                             |
| wuiiittttui   | DWORD | GetNamedSecurityInfoW(_In_ LPWSTR SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation *ppsidOwner, _Out_opt_ PSID *ppsidGroup *ppDacl, _Out_opt_ PACL *ppSacl, _Out PSECURITY_DESCRIPTOR *ppSecurity                                                              |
| tti           | BOOL  | GetNumberOfEventLogRecords(_In_ HANDLE _Out_ PDWORD NumberOfRecords)                                                                                                                                                                                                                   |
| tti           | BOOL  | GetOldestEventLogRecord(_In_ HANDLE PDWORD OldestRecord)                                                                                                                                                                                                                               |
| tuituiti      | BOOL  | GetPrivateObjectSecurity(_In_ PSECURITY_DESCRIPTOR ObjectDescriptor, _In_ SECURITY_INFORMATION SecurityInformation, _Out_opt_ PSECURITY_DESCRIPTOR                                                                                                                                     |

|             |       |                                                                                                                                                                                                                           |
|-------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |       | ResultantDescriptor, _In_ DWORD Des<br>PDWORD ReturnLength)                                                                                                                                                               |
| ttti        | BOOL  | GetSecurityDescriptorControl(_In_<br>PSECURITY_DESCRIPTOR pSecurity<br>PSECURITY_DESCRIPTOR_CONTRC<br>LPDWORD lpdwRevision)                                                                                               |
| ttti        | BOOL  | GetSecurityDescriptorDacl(_In_ PSECU<br>pSecurityDescriptor, _Out_ LPBOOL lpl<br>PACL *pDacl, _Out_ LPBOOL lpbDacl                                                                                                        |
| ttti        | BOOL  | GetSecurityDescriptorGroup(_In_ PSEC<br>pSecurityDescriptor, _Out_ PSID *pGro<br>lpbGroupDefaulted)                                                                                                                       |
| tui         | DWORD | GetSecurityDescriptorLength(_In_ PSEC<br>pSecurityDescriptor)                                                                                                                                                             |
| ttti        | BOOL  | GetSecurityDescriptorOwner(_In_ PSEC<br>pSecurityDescriptor, _Out_ PSID *pOw<br>lpbOwnerDefaulted)                                                                                                                        |
| ttui        | DWORD | GetSecurityDescriptorRMControl(_In_<br>PSECURITY_DESCRIPTOR SecurityD<br>PUCHAR RMControl)                                                                                                                                |
| ttti        | BOOL  | GetSecurityDescriptorSacl(_In_ PSECU<br>pSecurityDescriptor, _Out_ LPBOOL lpl<br>PACL *pSacl, _Out_ LPBOOL lpbSaclD                                                                                                       |
| tuiuittttui | DWORD | GetSecurityInfo(_In_ HANDLE handle,<br>SE_OBJECT_TYPE ObjectType, _In_<br>SECURITY_INFORMATION SecurityI<br>*ppsidOwner, _Out_opt_ PSID *ppsidG<br>*ppDacl, _Out_opt_ PACL *ppSacl, _Ou<br>PSECURITY_DESCRIPTOR *ppSecuri |
| tssti       | BOOL  | GetServiceDisplayName(_In_ SC_HAN<br>LPCTSTR lpServiceName, _Out_opt_ L<br>_Inout_ LPDWORD lpcchBuffer)                                                                                                                   |
| taati       | BOOL  | GetServiceDisplayNameA(_In_ SC_HA<br>_In_ LPCSTR lpServiceName, _Out_opt<br>lpDisplayName, _Inout_ LPDWORD lpc                                                                                                            |
| twwti       | BOOL  | GetServiceDisplayNameW(_In_ SC_HA<br>_In_ LPCWSTR lpServiceName, _Out_ o<br>lpDisplayName, _Inout_ LPDWORD lpc                                                                                                            |
| tssti       | BOOL  | GetServiceKeyName(_In_ SC_HANDLI<br>LPCTSTR lpDisplayName, _Out_opt_ L<br>_Inout_ LPDWORD lpcchBuffer)                                                                                                                    |
| taati       | BOOL  | GetServiceKeyNameA(_In_ SC_HAND<br>LPCSTR lpDisplayName, _Out_opt_ LP<br>_Inout_ LPDWORD lpcchBuffer)                                                                                                                     |
|             |       | GetServiceKeyNameW(_In_ SC_HAND                                                                                                                                                                                           |

|           |                           |                                                                                                                                                                                                 |
|-----------|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twwti     | BOOL                      | LPCWSTR lpDisplayName, _Out_opt_ LPDWORD lpServiceName, _Inout_ LPDWORD lpServiceName                                                                                                           |
| tt        | PSID_IDENTIFIER_AUTHORITY | GetSidIdentifierAuthority(_In_ PSID pSid)                                                                                                                                                       |
| ucui      | DWORD                     | GetSidLengthRequired(_In_ UCHAR nSidLength)                                                                                                                                                     |
| tuit      | PDWORD                    | GetSidSubAuthority(_In_ PSID pSid, _In_ DWORD nSubAuthority)                                                                                                                                    |
| tt        | PUCHAR                    | GetSidSubAuthorityCount(_In_ PSID pSid)                                                                                                                                                         |
| ttuiuitti | BOOL                      | GetThreadWaitChain(_In_ HWCT WctHandle, _In_ DWORD_PTR Context, _In_ DWORD ThreadId, _Inout_ LPDWORD NodeCount, _Inout_ PWAITCHAIN_NODE_INFO NodeInfo, _In_ BOOL IsCycle)                       |
| tuituiti  | BOOL                      | GetTokenInformation(_In_ HANDLE Token, _In_ TOKEN_INFORMATION_CLASS TokenInformationClass, _Out_opt_ LPVOID TokenInformation, _In_ DWORD TokenInformationLength, _Out_ PDWORD_PTR ReturnLength) |
| tui       | ULONG                     | GetTraceEnableFlags(_In_ TRACEHANDLE TraceHandle)                                                                                                                                               |
| tuc       | UCHAR                     | GetTraceEnableLevel(_In_ TRACEHANDLE TraceHandle)                                                                                                                                               |
| tt        | TRACEHANDLE               | GetTraceLoggerHandle(_In_ PVOID Buffer)                                                                                                                                                         |
| tt        | TRUSTEE_FORM              | GetTrusteeForm(_In_ PTRUSTEE pTrustee)                                                                                                                                                          |
| tt        | TRUSTEE_FORM              | GetTrusteeFormA(_In_ PTRUSTEE pTrustee)                                                                                                                                                         |
| tt        | TRUSTEE_FORM              | GetTrusteeFormW(_In_ PTRUSTEE pTrustee)                                                                                                                                                         |
| ts        | LPTSTR                    | GetTrusteeName(_In_ PTRUSTEE pTrustee)                                                                                                                                                          |
| ta        | LPSTR                     | GetTrusteeNameA(_In_ PTRUSTEE pTrustee)                                                                                                                                                         |
| tw        | LPWSTR                    | GetTrusteeNameW(_In_ PTRUSTEE pTrustee)                                                                                                                                                         |
| tui       | TRUSTEE_TYPE              | GetTrusteeType(_In_opt_ PTRUSTEE pTrustee)                                                                                                                                                      |
| tui       | TRUSTEE_TYPE              | GetTrusteeTypeA(_In_opt_ PTRUSTEE pTrustee)                                                                                                                                                     |
| tui       | TRUSTEE_TYPE              | GetTrusteeTypeW(_In_opt_ PTRUSTEE pTrustee)                                                                                                                                                     |
| sti       | BOOL                      | GetUserName(_Out_ LPTSTR lpBuffer, _In_ DWORD lpnSize)                                                                                                                                          |
| ati       | BOOL                      | GetUserNameA(_Out_ LPSTR lpBuffer, _In_ DWORD lpnSize)                                                                                                                                          |
| wti       | BOOL                      | GetUserNameW(_Out_ LPWSTR lpBuffer, _In_ DWORD lpnSize)                                                                                                                                         |
| ttti      | BOOL                      | GetWindowsAccountDomainSid(_In_ PSID pSid, _Out_ PSID ppDomainSid, _Inout_ DWORD *pnDomainSid)                                                                                                  |
| ti        | BOOL                      | ImpersonateAnonymousToken(_In_ HANDLE hToken)                                                                                                                                                   |
| ti        | BOOL                      | ImpersonateLoggedOnUser(_In_ HANDLE hToken)                                                                                                                                                     |
| ti        | BOOL                      | ImpersonateNamedPipeClient(_In_ HANDLE hToken)                                                                                                                                                  |

|           |       |                                                                                                                                                                                             |
|-----------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uii       | BOOL  | ImpersonateSelf(_In_ SECURITY_IMPERSONATION_LEVEL ImpersonationLevel)                                                                                                                       |
| tuiiii    | BOOL  | InitializeAcl(_Out_ PACL pAcl, _In_ DWORD dwAclRevision)                                                                                                                                    |
| tuii      | BOOL  | InitializeSecurityDescriptor(_Out_ PSECURITY_DESCRIPTOR pSecurityDescriptor, _In_ DWORD dwRevision)                                                                                         |
| ttuci     | BOOL  | InitializeSid(_Out_ PSID Sid, _In_ PSID_IDENTIFIER_AUTHORITY pIdentifierAuthority, _In_ BYTE nSubAuthorityCount)                                                                            |
| ssuiiiiii | DWORD | InitiateShutdown(_In_opt_ LPCTSTR lpMachineName, _In_ LPCTSTR lpMessage, _In_ DWORD dwGraceTime, _In_ DWORD dwShutdownFlags, _In_ DWORD dwTimeout)                                          |
| aauiiiiii | DWORD | InitiateShutdownA(_In_opt_ LPCTSTR lpMachineName, _In_ LPSTR lpMessage, _In_ DWORD dwGraceTime, _In_ DWORD dwShutdownFlags, _In_ DWORD dwTimeout)                                           |
| wwuiiiiii | DWORD | InitiateShutdownW(_In_opt_ LPWSTR lpMachineName, _In_ LPWSTR lpMessage, _In_ DWORD dwGraceTime, _In_ DWORD dwShutdownFlags, _In_ DWORD dwTimeout)                                           |
| ssuiiii   | BOOL  | InitiateSystemShutdown(_In_opt_ LPCTSTR lpMachineName, _In_opt_ LPCTSTR lpMessage, _In_ DWORD dwTimeout, _In_ BOOL bForceAppsClosed, _In_ BOOL bRebootAfterShutdown)                        |
| aauiiii   | BOOL  | InitiateSystemShutdownA(_In_opt_ LPCTSTR lpMachineName, _In_opt_ LPSTR lpMessage, _In_ DWORD dwTimeout, _In_ BOOL bForceAppsClosed, _In_ BOOL bRebootAfterShutdown)                         |
| ssuiiiiii | BOOL  | InitiateSystemShutdownEx(_In_opt_ LPCTSTR lpMachineName, _In_opt_ LPCTSTR lpMessage, _In_ DWORD dwTimeout, _In_ BOOL bForceAppsClosed, _In_ BOOL bRebootAfterShutdown, _In_ DWORD dwReason) |
| aauiiiiii | BOOL  | InitiateSystemShutdownExA(_In_opt_ LPCTSTR lpMachineName, _In_opt_ LPSTR lpMessage, _In_ DWORD dwTimeout, _In_ BOOL bForceAppsClosed, _In_ BOOL bRebootAfterShutdown, _In_ DWORD dwReason)  |
| wwuiiiiii | BOOL  | InitiateSystemShutdownExW(_In_opt_ LPCTSTR lpMachineName, _In_opt_ LPWSTR lpMessage, _In_ DWORD dwTimeout, _In_ BOOL bForceAppsClosed, _In_ BOOL bRebootAfterShutdown, _In_ DWORD dwReason) |
| wwuiiii   | BOOL  | InitiateSystemShutdownW(_In_opt_ LPCTSTR lpMachineName, _In_opt_ LPWSTR lpMessage, _In_ DWORD dwTimeout, _In_ BOOL bForceAppsClosed, _In_ BOOL bRebootAfterShutdown)                        |
| tui       | DWORD | InstallApplication(_In_ PINSTALLDATA pInstallData)                                                                                                                                          |
| titi      | BOOL  | IsTextUnicode(_In_ const VOID *lpv, _In_ LPINT lpiResult)                                                                                                                                   |
| ti        | BOOL  | IsTokenRestricted(_In_ HANDLE Token)                                                                                                                                                        |
| ti        | BOOL  | IsValidAcl(_In_ PACL pAcl)                                                                                                                                                                  |

|               |         |                                                                                                                                                                                                                                                                                                                                                |
|---------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti            | BOOL    | IsValidSecurityDescriptor(_In_ PSECURITY_DESCRIPTOR pSecurityDescriptor)                                                                                                                                                                                                                                                                       |
| ti            | BOOL    | IsValidSid(_In_ PSID pSid)                                                                                                                                                                                                                                                                                                                     |
| tuii          | BOOL    | IsWellKnownSid(_In_ PSID pSid, _In_ WELL_KNOWN_SID_TYPE WellKnownSidType)                                                                                                                                                                                                                                                                      |
| tt            | SC_LOCK | LockServiceDatabase(_In_ SC_HANDLE hService)                                                                                                                                                                                                                                                                                                   |
| sssuiuiti     | BOOL    | LogonUser(_In_ LPCTSTR lpszUsername, _In_ LPCTSTR lpszDomain, _In_opt_ LPCTSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _In_ PHANDLE phToken)                                                                                                                                                                            |
| aaaiiuiti     | BOOL    | LogonUserA(_In_ LPSTR lpszUsername, _In_ LPSTR lpszDomain, _In_opt_ LPSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _In_ PHANDLE phToken)                                                                                                                                                                                 |
| sssuiuitttti  | BOOL    | LogonUserEx(_In_ LPCTSTR lpszUsername, _In_ LPCTSTR lpszDomain, _In_opt_ LPCTSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _In_ PHANDLE phToken, _Out_opt_ PSID *pSid, _Out_opt_ PVOID *ppProfileBuffer, _Out_opt_ LPDWORD pdwProfileLength, _Out_opt_ PQUOTA_LIMIT *pQuotaLimit)                                         |
| aaaiiuitttti  | BOOL    | LogonUserExA(_In_ LPSTR lpszUsername, _In_ LPSTR lpszDomain, _In_opt_ LPSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _In_ PHANDLE phToken, _Out_opt_ PSID *pSid, _Out_opt_ PVOID *ppProfileBuffer, _Out_opt_ LPDWORD pdwProfileLength, _Out_opt_ PQUOTA_LIMIT *pQuotaLimit)                                              |
| sssuiuittttti | BOOL    | LogonUserExExW(_In_ LPCTSTR lpszUsername, _In_ LPCTSTR lpszDomain, _In_opt_ LPCTSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _In_opt_ PTOKEN_GROUPS pTokenGroups, _In_ PHANDLE phToken, _Out_opt_ PSID *pSid, _Out_opt_ PVOID *ppProfileBuffer, _Out_opt_ LPDWORD pdwProfileLength, _Out_opt_ PQUOTA_LIMIT *pQuotaLimit) |
| wwwuiuittttti | BOOL    | LogonUserExW(_In_ LPWSTR lpszUsername, _In_ LPWSTR lpszDomain, _In_opt_ LPWSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _Out_opt_ PHANDLE phToken, _Out_opt_ PVOID *ppProfileBuffer, _Out_opt_ LPDWORD pdwProfileLength, _Out_opt_ PQUOTA_LIMIT *pQuotaLimit)                                                            |
| wwwuiuiti     | BOOL    | LogonUserW(_In_ LPWSTR lpszUsername, _In_ LPWSTR lpszDomain, _In_opt_ LPWSTR lpszPassword, _In_ DWORD dwLogonType, _In_ DWORD dwLogonFlags, _In_ PHANDLE phToken)                                                                                                                                                                              |
| ssttsti       | BOOL    | LookupAccountName(_In_opt_ LPCTSTR lpAccountName, _Out_opt_ LPDWORD cbSid, _Out_opt_ LPCTSTR lpReferencedDomainName, _Inout_ LPDWORD pSid, _Out_opt_ LPCTSTR lpReferencedDomainName)                                                                                                                                                           |

|          |      |                                                                                                                                                                                        |
|----------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |      | cchReferencedDomainName, _Out_ PSI                                                                                                                                                     |
| aattatti | BOOL | LookupAccountNameA(_In_opt_ LPCSTR lpAccountName, _Out_opt_ PSID lpSid, _Out_opt_ LPSTR RchReferencedDomainName, _Inout_ LPDWORD cchReferencedDomainName, _Out_ PSID_NAME_USE peUse)   |
| wwttwti  | BOOL | LookupAccountNameW(_In_opt_ LPCWSTR lpAccountName, _Out_opt_ PSID lpSid, _Out_opt_ LPWSTR RchReferencedDomainName, _Inout_ LPDWORD cchReferencedDomainName, _Out_ PSID_NAME_USE peUse) |
| stststi  | BOOL | LookupAccountSid(_In_opt_ LPCTSTR lpSid, _Out_opt_ LPTSTR lpName, _Out_opt_ LPSTR lpReferencedDomainName, _Inout_ LPDWORD cchReferencedDomainName, _Out_ PSID_NAME_USE peUse)          |
| atatatti | BOOL | LookupAccountSidA(_In_opt_ LPCSTR lpSid, _Out_opt_ LPSTR lpName, _Out_opt_ LPSTR lpReferencedDomainName, _Inout_ LPDWORD cchReferencedDomainName, _Out_ PSID_NAME_USE peUse)           |
| wtwttwti | BOOL | LookupAccountSidW(_In_opt_ LPCWSTR lpSid, _Out_opt_ LPWSTR lpName, _Out_opt_ LPWSTR lpReferencedDomainName, _Inout_ LPDWORD cchReferencedDomainName, _Out_ PSID_NAME_USE peUse)        |
| sssti    | BOOL | LookupPrivilegeDisplayName(_In_opt_ LPCTSTR lpSystemName, _In_ LPCTSTR lpName, _Out_ LPSTR lpDisplayName, _Inout_ LPDWORD cchDisplayName, _Inout_ LPDWORD lpLanguageId)                |
| aaatti   | BOOL | LookupPrivilegeDisplayNameA(_In_opt_ LPCTSTR lpSystemName, _In_ LPCSTR lpName, _Out_ LPSTR lpDisplayName, _Inout_ LPDWORD cchDisplayName, _Inout_ LPDWORD lpLanguageId)                |
| wwwtti   | BOOL | LookupPrivilegeDisplayNameW(_In_opt_ LPCTSTR lpSystemName, _In_ LPCWSTR lpName, _Out_ LPWSTR lpDisplayName, _Inout_ LPDWORD cchDisplayName, _Inout_ LPDWORD lpLanguageId)              |
| ststi    | BOOL | LookupPrivilegeName(_In_opt_ LPCTSTR lpLuid, _Out_opt_ LPTSTR lpName, _Out_ LPSTR lpName, _Out_ LPDWORD cchName)                                                                       |
| atati    | BOOL | LookupPrivilegeNameA(_In_opt_ LPCSTR lpLuid, _Out_opt_ LPSTR lpName, _Out_ LPSTR lpName, _Out_ LPDWORD cchName)                                                                        |
| wtwti    | BOOL | LookupPrivilegeNameW(_In_opt_ LPCWSTR lpLuid, _Out_opt_ LPWSTR lpName, _Out_ LPWSTR lpName, _Out_ LPDWORD cchName)                                                                     |

|          |       |                                                                                                                                                                                                                                                                                |
|----------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ssti     | BOOL  | LookupPrivilegeValue(_In_opt_ LPCTSTR lpName, _Out_ PLUID lpLuid)                                                                                                                                                                                                              |
| aati     | BOOL  | LookupPrivilegeValueA(_In_opt_ LPCSTR lpName, _Out_ PLUID lpLuid)                                                                                                                                                                                                              |
| wwti     | BOOL  | LookupPrivilegeValueW(_In_opt_ LPCWSTR lpName, _Out_ PLUID lpLuid)                                                                                                                                                                                                             |
| ttttttui | DWORD | LookupSecurityDescriptorParts(_Out_opt_ POWNER *pOwner, _Out_opt_ PTRUSTEE *pGrantee, PULONG cCountOfAccessEntries, _Out_opt_ PEXPLICIT_ACCESS *pListOfAccessEntries, PULONG cCountOfAuditEntries, _Out_opt_ PEXPLICIT_ACCESS *pListOfAuditEntries, PSECURITY_DESCRIPTOR pSD)  |
| ttttttui | DWORD | LookupSecurityDescriptorPartsA(_Out_opt_ POWNER *pOwner, _Out_opt_ PTRUSTEE *pGrantee, PULONG cCountOfAccessEntries, _Out_opt_ PEXPLICIT_ACCESS *pListOfAccessEntries, PULONG cCountOfAuditEntries, _Out_opt_ PEXPLICIT_ACCESS *pListOfAuditEntries, PSECURITY_DESCRIPTOR pSD) |
| ttttttui | DWORD | LookupSecurityDescriptorPartsW(_Out_opt_ POWNER *pOwner, _Out_opt_ PTRUSTEE *pGrantee, PULONG cCountOfAccessEntries, _Out_opt_ PEXPLICIT_ACCESS *pListOfAccessEntries, PULONG cCountOfAuditEntries, _Out_opt_ PEXPLICIT_ACCESS *pListOfAuditEntries, PSECURITY_DESCRIPTOR pSD) |
| ttsuii   | int   | LsaAddAccountRights(_In_ LSA_HANDLE PolicyHandle, _In_ PSID AccountSid, _In_ PLSA_UNICODE_STRING *UserRights, _In_ ULONG CountOfRights)                                                                                                                                        |
| ti       | int   | LsaClose(_In_ LSA_HANDLE ObjectHandle)                                                                                                                                                                                                                                         |
| tttuiti  | int   | LsaCreateTrustedDomainEx(_In_ LSA_HANDLE PolicyHandle, _In_ PTRUSTED_DOMAIN_INFORMATION *TrustedDomainInformation, _In_ PTRUSTED_DOMAIN_AUTH_INFORMATION *AuthenticationInformation, _In_ ACCESS_MASK DesiredAccess, _Out_ PLSA_HANDLE *NewHandle)                             |
| titi     | int   | LsaDeleteTrustedDomain(_In_ LSA_HANDLE PolicyHandle, _In_ PSID TrustedDomainSid)                                                                                                                                                                                               |
| tttiti   | int   | LsaEnumerateAccountRights(_In_ LSA_HANDLE PolicyHandle, _In_ PSID AccountSid, _Out_ PLSA_UNICODE_STRING *UserRights, _Out_ PULONG CountOfRights)                                                                                                                               |
| tssti    | int   | LsaEnumerateAccountsWithUserRights(_In_ LSA_HANDLE PolicyHandle, _In_ PLSA_UNICODE_STRING *UserRights, _Out_ PVOID *EnumerationBuffer, _Out_ PULONG CountOfAccounts)                                                                                                           |

|          |       |                                                                                                                                                                                                             |
|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |       | CountReturned)                                                                                                                                                                                              |
| tttuiti  | int   | LsaEnumerateTrustedDomains(_In_ LSA_POLICY_HANDLE PolicyHandle, _In_ PLSA_ENUMERATION_CONTEXT EnumerationContext, _Out_ PVOID *Buffer, _In_ ULONG PreferredMaximumLength, _Out_ PULONG CountReturned)       |
| tttuiti  | int   | LsaEnumerateTrustedDomainsEx(_In_ LSA_POLICY_HANDLE PolicyHandle, _In_ PLSA_ENUMERATION_CONTEXT EnumerationContext, _Out_ PVOID *Buffer, _In_ ULONG PreferredMaximumLength, _Out_ PULONG CountReturned)     |
| ti       | int   | LsaFreeMemory(_In_ PVOID Buffer)                                                                                                                                                                            |
| tuiusti  | int   | LsaLookupNames(_In_ LSA_HANDLE PolicyHandle, _In_ ULONG Count, _In_ PLSA_UNICODE_STRING Names, _Out_ PLSA_REFERENCED_DOMAIN_LIST ReferencedDomainList, _Out_ PLSA_TRANSLATED_SID *Sids)                     |
| tuiiusti | int   | LsaLookupNames2(_In_ LSA_HANDLE PolicyHandle, _In_ ULONG Flags, _In_ ULONG Count, _In_ PLSA_UNICODE_STRING Names, _Out_ PLSA_REFERENCED_DOMAIN_LIST ReferencedDomainList, _Out_ PLSA_TRANSLATED_SID2 *Sids) |
| twti     | int   | LsaLookupPrivilegeValue(LSA_HANDLE PolicyHandle, PUNICODE_STRING Name, PLUID Value)                                                                                                                         |
| tuittti  | int   | LsaLookupSids(_In_ LSA_HANDLE PolicyHandle, _In_ ULONG Count, _In_ PSID *Sids, _Out_ PLSA_REFERENCED_DOMAIN_LIST ReferencedDomainList, _Out_ PLSA_TRANSLATED_NAME *Names)                                   |
| uiui     | ULONG | LsaNtStatusToWinError(_In_ NTSTATUS Status)                                                                                                                                                                 |
| stuiti   | int   | LsaOpenPolicy(_In_ PLSA_UNICODE_STRING PolicyName, _In_ PLSA_OBJECT_ATTRIBUTES ObjectAttributes, _In_ ACCESS_MASK DesiredAccess, _Inout_ PLSA_HANDLE PolicyHandle)                                          |
| tsuiti   | int   | LsaOpenTrustedDomainByName(_In_ LSA_POLICY_HANDLE PolicyHandle, _In_ PLSA_UNICODE_STRING TrustedDomainName, _In_ ACCESS_MASK DesiredAccess, _Out_ PLSA_HANDLE TrustedDomainHandle)                          |
| tuiti    | int   | LsaQueryDomainInformationPolicy(_In_ LSA_POLICY_HANDLE PolicyHandle, _In_ PLSA_UNICODE_STRING PolicyName, _In_ POLICY_DOMAIN_INFORMATION_CLASS InformationClass, _Out_ PVOID *Buffer)                       |
| tsti     | int   | LsaQueryForestTrustInformation(_In_ LSA_POLICY_HANDLE PolicyHandle, _In_ PLSA_UNICODE_STRING TrustedDomainName, _Out_ PLSA_FOREST_TRUST_INFORMATION *TrustInformation)                                      |
| tuiti    | int   | LsaQueryInformationPolicy(_In_ LSA_POLICY_HANDLE PolicyHandle, _In_ POLICY_INFORMATION_CLASS InformationClass, _Out_ PVOID *Buffer)                                                                         |

|            |      |                                                                                                                                                                                                                                               |
|------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiti     | int  | LsaQueryTrustedDomainInfo(_In_ LSA_PolicyHandle, _In_ PSID TrustedDomainName, _In_ TRUSTED_INFORMATION_CLASS InformationClass, _In_ PVOID *Buffer)                                                                                            |
| tsuiti     | int  | LsaQueryTrustedDomainInfoByName(_In_ LSA_PolicyHandle, _In_ PLSA_UNICODENAME TrustedDomainName, _In_ TRUSTED_INFORMATION_CLASS InformationClass, _In_ PVOID *Buffer)                                                                          |
| ttucsuii   | int  | LsaRemoveAccountRights(_In_ LSA_HANDLE PolicyHandle, _In_ PSID AccountSid, _In_ BOOLEAN RemoveAllRights, _In_ PLSA_UNICODENAME UserRights, _In_ CountOfRights)                                                                                |
| tsti       | int  | LsaRetrievePrivateData(_In_ LSA_HANDLE PolicyHandle, _In_ PLSA_UNICODENAME KeyName, _In_ PLSA_UNICODENAME *PrivateData)                                                                                                                       |
| tuiti      | int  | LsaSetDomainInformationPolicy(_In_ LSA_HANDLE PolicyHandle, _In_ POLICY_DOMAIN_INFORMATION_CLASS InformationClass, _In_ PVOID Buffer)                                                                                                         |
| tstucti    | int  | LsaSetForestTrustInformation(_In_ LSA_HANDLE PolicyHandle, _In_ PLSA_UNICODENAME TrustedDomainName, _In_ PLSA_FOREST_TRUST_INFORMATION ForestTrustInfo, _In_ BOOLEAN CheckOnly, _Out_ PLSA_FOREST_TRUST_COLLISION_INFORMATION *CollisionInfo) |
| tuiti      | int  | LsaSetInformationPolicy(_In_ LSA_HANDLE PolicyHandle, _In_ POLICY_INFORMATION_CLASS InformationClass, _In_ PVOID Buffer)                                                                                                                      |
| tsuiti     | int  | LsaSetTrustedDomainInfoByName(_In_ LSA_HANDLE PolicyHandle, _In_ PLSA_UNICODENAME TrustedDomainName, _In_ TRUSTED_INFORMATION_CLASS InformationClass, _In_ PVOID Buffer)                                                                      |
| ttuiti     | int  | LsaSetTrustedDomainInformation(_In_ LSA_HANDLE PolicyHandle, _In_ PSID TrustedDomainName, _In_ TRUSTED_INFORMATION_CLASS InformationClass, _In_ PVOID Buffer)                                                                                 |
| tssi       | int  | LsaStorePrivateData(_In_ LSA_HANDLE PolicyHandle, _In_ PLSA_UNICODENAME KeyName, _In_ PLSA_UNICODENAME PrivateData)                                                                                                                           |
| ttttttttti | BOOL | MakeAbsoluteSD(_In_ PSECURITY_DESCRIPTOR pSelfRelativeSD, _Out_opt_ PSECURITY_DESCRIPTOR pAbsoluteSD, _Inout_ LPDWORD lpdwDacl, _Out_opt_ PACL pDacl, _Inout_ LPDWORD lpdwSacl, _Out_opt_ PACL pSacl, _Inout_ LPDWORD lpdwDacl)               |

|            |       |                                                                                                                                                                                                                                                                                                                           |
|------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |       | <code>_Out_opt_PSID pOwner, _Inout_LPVOID pPrimaryGroup, _Inout_LPVOID pPrimaryGroupSize)</code>                                                                                                                                                                                                                          |
| tti        | BOOL  | <code>MakeSelfRelativeSD(_In_PSECURITY_DESCRIPTOR pAbsoluteSD, _Out_opt_PSECURITY_DESCRIPTOR pSelfRelativeSD, _Inout_LPVOID lp</code>                                                                                                                                                                                     |
| tii        | VOID  | <code>MapGenericMask(_Inout_PDWORD_ARRAY pGenericMapping GenericMapping</code>                                                                                                                                                                                                                                            |
| ttuctttui  | DWORD | <code>MSChapSvcChangePassword(_In_PWSTR UserN</code><br><code>PLM_OWF_PASSWORD LmOldOwfPassword</code><br><code>PLM_OWF_PASSWORD LmNewOwfPassword</code><br><code>PNT_OWF_PASSWORD NtOldOwfPassword</code><br><code>PNT_OWF_PASSWORD NtNewOwfPassword</code>                                                              |
| ttttucttui | DWORD | <code>MSChapSvcChangePassword2(_In_PWSTR UserN</code><br><code>PSAMPR_ENCRYPTED_USER_PASSWORD NewPasswordEncryptedWithOldNt, _In_PSECURITY_DESCRIPTOR OldNtOwfPasswordEncryptedWithNewLmPresent, _In_PSECURITY_DESCRIPTOR NewPasswordEncryptedWithOldLm, _In_PSECURITY_DESCRIPTOR OldLmOwfPasswordEncryptedWithNew</code> |
| ii         | BOOL  | <code>NotifyBootConfigStatus(_In_BOOL Boot</code>                                                                                                                                                                                                                                                                         |
| tii        | BOOL  | <code>NotifyChangeEventLog(_In_HANDLE_EVENT HANDLE hEvent)</code>                                                                                                                                                                                                                                                         |
| tuitui     | DWORD | <code>NotifyServiceStatusChange(_In_SERVICE_STATUS_HANDLE dwNotifyMask, _In_PSERVICE_STATUS pNotifyBuffer)</code>                                                                                                                                                                                                         |
| tuitui     | DWORD | <code>NotifyServiceStatusChangeA(_In_SERVICE_STATUS_HANDLE dwNotifyMask, _In_PSERVICE_STATUS pNotifyBuffer)</code>                                                                                                                                                                                                        |
| tuitui     | DWORD | <code>NotifyServiceStatusChangeW(_In_SERVICE_STATUS_HANDLE dwNotifyMask, _In_PSERVICE_STATUS pNotifyBuffer)</code>                                                                                                                                                                                                        |
| stii       | BOOL  | <code>ObjectCloseAuditAlarm(_In_LPCTSTR ObjectName, LPVOID HandleId, _In_BOOL Generate</code>                                                                                                                                                                                                                             |
| atii       | BOOL  | <code>ObjectCloseAuditAlarmA(_In_LPCTSTR ObjectName, LPVOID HandleId, _In_BOOL Generate</code>                                                                                                                                                                                                                            |
| wtii       | BOOL  | <code>ObjectCloseAuditAlarmW(_In_LPCTSTR ObjectName, _In_LPVOID HandleId, _In_BOOL Generate</code>                                                                                                                                                                                                                        |
| stii       | BOOL  | <code>ObjectDeleteAuditAlarm(_In_LPCTSTR ObjectName, LPVOID HandleId, _In_BOOL Generate</code>                                                                                                                                                                                                                            |
| atii       | BOOL  | <code>ObjectDeleteAuditAlarmA(_In_LPCTSTR ObjectName, LPVOID HandleId, _In_BOOL Generate</code>                                                                                                                                                                                                                           |

|               |        |                                                                                                                                                                                                                                                                                                       |
|---------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |        | LPVOID HandleId, _In_ BOOL Generat                                                                                                                                                                                                                                                                    |
| wtii          | BOOL   | ObjectDeleteAuditAlarmW(_In_ LPCW<br>_In_ LPVOID HandleId, _In_ BOOL Ge                                                                                                                                                                                                                               |
| stssttuiuitti | BOOL   | ObjectOpenAuditAlarm(_In_ LPCTSTR<br>LPVOID HandleId, _In_ LPTSTR Objec<br>LPTSTR ObjectName, _In_ PSECURIT<br>pSecurityDescriptor, _In_ HANDLE Cli<br>DesiredAccess, _In_ DWORD GrantedA<br>PPRIVILEGE_SET Privileges, _In_ BO<br>BOOL AccessGranted, _Out_ LPBOOL                                   |
| ataattuiuitti | BOOL   | ObjectOpenAuditAlarmA(_In_ LPCSTR<br>LPVOID HandleId, _In_ LPSTR Objec<br>LPSTR ObjectName, _In_ PSECURITY<br>pSecurityDescriptor, _In_ HANDLE Cli<br>DesiredAccess, _In_ DWORD GrantedA<br>PPRIVILEGE_SET Privileges, _In_ BO<br>BOOL AccessGranted, _Out_ LPBOOL                                    |
| wtwwtuiuitti  | BOOL   | ObjectOpenAuditAlarmW(_In_ LPCWS<br>_In_ LPVOID HandleId, _In_ LPWSTR<br>_In_opt_ LPWSTR ObjectName, _In_<br>PSECURITY_DESCRIPTOR pSecurity<br>HANDLE ClientToken, _In_ DWORD I<br>DWORD GrantedAccess, _In_opt_ PPR<br>Privileges, _In_ BOOL ObjectCreation, _<br>AccessGranted, _Out_ LPBOOL Genera |
| sttuitii      | BOOL   | ObjectPrivilegeAuditAlarm(_In_ LPCTS<br>_In_ LPVOID HandleId, _In_ HANDLE<br>DWORD DesiredAccess, _In_ PPRIVIL<br>_In_ BOOL AccessGranted)                                                                                                                                                            |
| attuitii      | BOOL   | ObjectPrivilegeAuditAlarmA(_In_ LPCS<br>_In_ LPVOID HandleId, _In_ HANDLE<br>DWORD DesiredAccess, _In_ PPRIVIL<br>_In_ BOOL AccessGranted)                                                                                                                                                            |
| wttuitii      | BOOL   | ObjectPrivilegeAuditAlarmW(_In_ LPC<br>_In_ LPVOID HandleId, _In_ HANDLE<br>DWORD DesiredAccess, _In_ PPRIVIL<br>_In_ BOOL AccessGranted)                                                                                                                                                             |
| sst           | HANDLE | OpenBackupEventLog(_In_ LPCTSTR I<br>LPCTSTR lpFileName)                                                                                                                                                                                                                                              |
| aat           | HANDLE | OpenBackupEventLogA(_In_ LPCSTR<br>LPCSTR lpFileName)                                                                                                                                                                                                                                                 |
| wwt           | HANDLE | OpenBackupEventLogW(_In_ LPCWST<br>_In_ LPCWSTR lpFileName)                                                                                                                                                                                                                                           |
| suitui        | DWORD  | OpenEncryptedFileRaw(_In_ LPCTSTR<br>ULONG ulFlags, _Out_ PVOID *pvCor                                                                                                                                                                                                                                |
| autui         | DWORD  | OpenEncryptedFileRawA(_In_ LPCSTR                                                                                                                                                                                                                                                                     |

|           |                           |                                                                                                             |
|-----------|---------------------------|-------------------------------------------------------------------------------------------------------------|
|           |                           | ULONG ulFlags, _Out_ PVOID *pvCon                                                                           |
| wuitui    | DWORD                     | OpenEncryptedFileRawW(_In_ LPCWS<br>ULONG ulFlags, _Out_ PVOID *pvCon                                       |
| sst       | HANDLE                    | OpenEventLog(_In_ LPCTSTR lpUNCS<br>LPCTSTR lpSourceName)                                                   |
| aat       | HANDLE                    | OpenEventLogA(_In_ LPCSTR lpUNCS<br>LPCSTR lpSourceName)                                                    |
| wwt       | HANDLE                    | OpenEventLogW(_In_ LPCWSTR lpUN<br>LPCWSTR lpSourceName)                                                    |
| tuiti     | BOOL                      | OpenProcessToken(_In_ HANDLE Proc<br>DWORD DesiredAccess, _Out_ PHANI                                       |
| ssuit     | SC_HANDLE                 | OpenSCManager(_In_opt_ LPCTSTR lp<br>LPCTSTR lpDatabaseName, _In_ DWO                                       |
| aauii     | SC_HANDLE                 | OpenSCManagerA(_In_opt_ LPCSTR lp<br>_In_opt_ LPCSTR lpDatabaseName, _In<br>dwDesiredAccess)                |
| wwuit     | SC_HANDLE                 | OpenSCManagerW(_In_opt_ LPCWSTR<br>_In_opt_ LPCWSTR lpDatabaseName, _<br>dwDesiredAccess)                   |
| tsuit     | SC_HANDLE                 | OpenService(_In_ SC_HANDLE hSCM<br>lpServiceName, _In_ DWORD dwDesir                                        |
| tauit     | SC_HANDLE                 | OpenServiceA(_In_ SC_HANDLE hSC<br>lpServiceName, _In_ DWORD dwDesir                                        |
| twuit     | SC_HANDLE                 | OpenServiceW(_In_ SC_HANDLE hSC<br>LPCWSTR lpServiceName, _In_ DWO                                          |
| tuiiti    | BOOL                      | OpenThreadToken(_In_ HANDLE Threa<br>DesiredAccess, _In_ BOOL OpenAsSelf<br>TokenHandle)                    |
| uitt      | HWCT                      | OpenThreadWaitChainSession(_In_ DW<br>PWAITCHAINCALLBACK callback)                                          |
| ttwuit    | PPERF_COUNTERSET_INSTANCE | PerfCreateInstance(_In_ HANDLE hPro<br>CounterSetGuid, _In_ LPCWSTR szInst<br>ULONG dwInstance)             |
| ttuiuiui  | ULONG                     | PerfDecrementULongCounterValue(_In_<br>_In_ PPERF_COUNTERSET_INSTANC<br>ULONG CounterId, _In_ ULONG IValu   |
| ttuiui6ui | ULONG                     | PerfDecrementULongLongCounterValue<br>hProvider, _In_ PPERF_COUNTERSET<br>_In_ ULONG CounterId, _In_ ULONGI |
| ttui      | ULONG                     | PerfDeleteInstance(_In_ HANDLE hPro<br>PPERF_COUNTERSET_INSTANCE In                                         |
| ttuiuiui  | ULONG                     | PerfIncrementULongCounterValue(_In_<br>_In_ PPERF_COUNTERSET_INSTANC                                        |

|           |                           |                                                                                                                                                                                                      |
|-----------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |                           | ULONG CounterId, _In_ ULONG IValue)                                                                                                                                                                  |
| ttuiui6ui | ULONG                     | PerfIncrementULongLongCounterValue(hProvider, _In_ PPERF_COUNTERSET_INSTANCE CounterSet, _In_ ULONG CounterId, _In_ ULONG IValue)                                                                    |
| ttwuit    | PPERF_COUNTERSET_INSTANCE | PerfQueryInstance(_In_ HANDLE hProvider, _In_ GUID CounterSetGuid, _In_ LPCWSTR szInstanceName, _In_ DWORD dwInstance)                                                                               |
| ttuitui   | ULONG                     | PerfSetCounterRefValue(_In_ HANDLE hProvider, _In_ PPERF_COUNTERSET_INSTANCE CounterSet, _In_ ULONG CounterId, _In_ PVOID lpAddr)                                                                    |
| ttuiui    | ULONG                     | PerfSetCounterSetInfo(_In_ HANDLE hProvider, _In_ PPERF_COUNTERSET_INSTANCE CounterSet, _In_ PPERF_COUNTERSET_INFO pTemplate, _In_ DWORD dwTemplateSize)                                             |
| ttuiuiui  | ULONG                     | PerfSetULongCounterValue(_In_ HANDLE hProvider, _In_ PPERF_COUNTERSET_INSTANCE CounterSet, _In_ ULONG CounterId, _In_ ULONG IValue)                                                                  |
| ttuiui6ui | ULONG                     | PerfSetULongLongCounterValue(_In_ HANDLE hProvider, _In_ PPERF_COUNTERSET_INSTANCE CounterSet, _In_ ULONG CounterId, _In_ ULONGLONG IValue)                                                          |
| tttui     | ULONG                     | PerfStartProvider(_In_ LPGUID ProviderGuid, _In_ PERFLIBREQUEST ControlCallback, _In_ HANDLE *phProvider)                                                                                            |
| tttui     | ULONG                     | PerfStartProviderEx(_In_ LPGUID ProviderGuid, _In_ PPERF_PROVIDER_CONTEXT ProviderContext, _In_ HANDLE *phProvider)                                                                                  |
| tui       | ULONG                     | PerfStopProvider(_In_ HANDLE hProvider)                                                                                                                                                              |
| ttti      | BOOL                      | PrivilegeCheck(_In_ HANDLE ClientToken, _In_ PPRIVILEGE_SET RequiredPrivileges, _Out_ PBOOLEAN *pResult)                                                                                             |
| sstii     | BOOL                      | PrivilegedServiceAuditAlarm(_In_ LPCWSTR ServiceName, _In_ HANDLE ClientToken, _In_ PPRIVILEGE_SET Privileges, _In_ BOOLEAN AuditSuccess)                                                            |
| aattii    | BOOL                      | PrivilegedServiceAuditAlarmA(_In_ LPCTSTR ServiceName, _In_ HANDLE ClientToken, _In_ PPRIVILEGE_SET Privileges, _In_ BOOLEAN AuditSuccess)                                                           |
| wwttii    | BOOL                      | PrivilegedServiceAuditAlarmW(_In_ LPWSTR SubsystemName, _In_ LPCWSTR ServiceName, _In_ HANDLE ClientToken, _In_ PPRIVILEGE_SET Privileges, _In_ BOOLEAN AuditSuccess, _In_ PBOOLEAN *pAccessGranted) |
| wtui      | DWORD                     | QueryRecoveryAgentsOnEncryptedFile(_In_ LPCTSTR lpFileName, _Out_ PULONG *pRecoveryAgents)                                                                                                           |
| uiti      | VOID                      | QuerySecurityAccessMask(_In_ SECURITY_INFORMATION SecurityInformation, _Out_ LPDWORD *pAccessMask)                                                                                                   |

|          |       |                                                                                                                                                                                                |
|----------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiti   | BOOL  | QueryServiceConfig(_In_ SC_HANDLE LPQUERY_SERVICE_CONFIG lpServiceName, _In_ DWORD cbBufSize, _Out_ LPDWORD)                                                                                   |
| tuituiti | BOOL  | QueryServiceConfig2(_In_ SC_HANDLE hService, _In_ DWORD dwInfoLevel, _Out_opt_ LPBYTE lpBuffer, _In_ DWORD cbBufSize, _Out_ LPDWORD)                                                           |
| tuituiti | BOOL  | QueryServiceConfig2A(_In_ SC_HANDLE hService, _In_ DWORD dwInfoLevel, _Out_opt_ LPBYTE lpBuffer, _In_ DWORD cbBufSize, _Out_ LPDWORD)                                                          |
| tuituiti | BOOL  | QueryServiceConfig2W(_In_ SC_HANDLE hService, _In_ DWORD dwInfoLevel, _Out_opt_ LPBYTE lpBuffer, _In_ DWORD cbBufSize, _Out_ LPDWORD)                                                          |
| ttuiti   | BOOL  | QueryServiceConfigA(_In_ SC_HANDLE hService, _In_ LPQUERY_SERVICE_CONFIG lpServiceName, _In_ DWORD cbBufSize, _Out_ LPDWORD)                                                                   |
| ttuiti   | BOOL  | QueryServiceConfigW(_In_ SC_HANDLE hService, _In_ LPQUERY_SERVICE_CONFIG lpServiceName, _In_ DWORD cbBufSize, _Out_ LPDWORD)                                                                   |
| ttuiti   | BOOL  | QueryServiceLockStatus(_In_ SC_HANDLE hService, _Out_opt_ LPQUERY_SERVICE_LOCK_STATUS lpLockStatus, _In_ DWORD cbBufSize, _Out_ pcbBytesNeeded)                                                |
| ttuiti   | BOOL  | QueryServiceLockStatusA(_In_ SC_HANDLE hService, _Out_opt_ LPQUERY_SERVICE_LOCK_STATUS lpLockStatus, _In_ DWORD cbBufSize, _Out_ pcbBytesNeeded)                                               |
| ttuiti   | BOOL  | QueryServiceLockStatusW(_In_ SC_HANDLE hService, _Out_opt_ LPQUERY_SERVICE_LOCK_STATUS lpLockStatus, _In_ DWORD cbBufSize, _Out_ pcbBytesNeeded)                                               |
| tuituiti | BOOL  | QueryServiceObjectSecurity(_In_ SC_HANDLE hService, _In_ SECURITY_INFORMATION dwSecurityInformation, _Out_opt_ PSECURITY_DESCRIPTOR lpSecurityDescriptor, _In_ DWORD cbBufSize, _Out_ LPDWORD) |
| tti      | BOOL  | QueryServiceStatus(_In_ SC_HANDLE hService, _Out_ LPSERVICE_STATUS lpServiceStatus)                                                                                                            |
| tuituiti | BOOL  | QueryServiceStatusEx(_In_ SC_HANDLE hService, _In_ SC_STATUS_TYPE InfoLevel, _Out_opt_ LPQUERY_SERVICE_STATUS lpServiceStatus, _In_ DWORD cbBufSize, _Out_ LPDWORD)                            |
| tstui    | ULONG | QueryTrace(_In_ TRACEHANDLE SessionName, _Inout_ PEVENT_TRACE_PROPERTIES Properties, _In_ ULONG SessionId)                                                                                     |
| tatui    | ULONG | QueryTraceA(_In_ TRACEHANDLE SessionName, _Inout_ LPCSTR SessionName, _Inout_ PEVENT_TRACE_PROPERTIES Properties, _In_ ULONG SessionId)                                                        |

|                |       |                                                                                                                                                                                                                          |
|----------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twtui          | ULONG | QueryTraceW(_In_ TRACEHANDLE SessionName, _Inout_ PEVENT_TRACE_PROPERTIES Properties)                                                                                                                                    |
| wtui           | DWORD | QueryUsersOnEncryptedFile(_In_ LPCWSTR FileName, _Out_ PENCIPHERING_CERTIFICATE_INFORMATION CertificateInformation)                                                                                                      |
| tttui          | DWORD | ReadEncryptedFileRaw(_In_ PFILE_EXPLIST FileList, _In_opt_ PVOID pvContext, _In_opt_ PFILE_EXPLIST_CALLBACK pfExportCallback, _In_opt_ PVOID pvContext)                                                                  |
| tuiuituitti    | BOOL  | ReadEventLog(_In_ HANDLE hEventLog, _In_ DWORD dwReadFlags, _In_ DWORD dwRecordId, _In_ LPVOID lpBuffer, _In_ DWORD nNumberOfBytesToRead, _Out_ DWORD *pnBytesRead, _Out_ DWORD *pnMinNumberOfBytesNeeded)               |
| tuiuituitti    | BOOL  | ReadEventLogA(_In_ HANDLE hEventLog, _In_ DWORD dwReadFlags, _In_ DWORD dwRecordId, _In_ LPVOID lpBuffer, _In_ DWORD nNumberOfBytesToRead, _Out_ DWORD *pnBytesRead, _Out_ DWORD *pnMinNumberOfBytesNeeded)              |
| tuiuituitti    | BOOL  | ReadEventLogW(_In_ HANDLE hEventLog, _In_ DWORD dwReadFlags, _In_ DWORD dwRecordId, _In_ LPVOID lpBuffer, _In_ DWORD nNumberOfBytesToRead, _Out_ DWORD *pnBytesRead, _Out_ DWORD *pnMinNumberOfBytesNeeded)              |
| tui            | LONG  | RegCloseKey(_In_ HKEY hKey)                                                                                                                                                                                              |
| sttui          | LONG  | RegConnectRegistry(_In_opt_ LPCTSTR lpSourcePath, _In_ HKEY hKey, _Out_ PHKEY phkResult)                                                                                                                                 |
| attui          | LONG  | RegConnectRegistryA(_In_opt_ LPCSTR lpSourcePath, _In_ HKEY hKey, _Out_ PHKEY phkResult)                                                                                                                                 |
| wttui          | LONG  | RegConnectRegistryW(_In_opt_ LPCWSTR lpSourcePath, _In_ HKEY hKey, _Out_ PHKEY phkResult)                                                                                                                                |
| tstui          | LONG  | RegCopyTree(_In_ HKEY hKeySrc, _In_ LPCTSTR lpSubKey, _In_ HKEY hKeyDest)                                                                                                                                                |
| tatui          | LONG  | RegCopyTreeA(_In_ HKEY hKeySrc, _In_ LPCTSTR lpSubKey, _In_ HKEY hKeyDest)                                                                                                                                               |
| twtui          | LONG  | RegCopyTreeW(_In_ HKEY hKeySrc, _In_ LPCTSTR lpSubKey, _In_ HKEY hKeyDest)                                                                                                                                               |
| tstui          | LONG  | RegCreateKey(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _Out_ PHKEY phkResult)                                                                                                                                           |
| tatui          | LONG  | RegCreateKeyA(_In_ HKEY hKey, _In_opt_ LPCSTR lpSubKey, _Out_ PHKEY phkResult)                                                                                                                                           |
| tsuisuiuitttui | LONG  | RegCreateKeyEx(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwOptions, _In_opt_ REGSAM samDesired, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult, _Out_opt_ LPDWORD lpDisposition) |

|                 |      |                                                                                                                                                                                                                                                                                                           |
|-----------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tauiauiuitttui  | LONG | RegCreateKeyExA(_In_ HKEY hKey, _Reserved_ DWORD Reserved, _In_opt_ DWORD dwOptions, _In_ REGSAM samDesired, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult, _Out_opt_ LPDWORD lpdwDisposition)                                                                               |
| twuiwuiuitttui  | LONG | RegCreateKeyExW(_In_ HKEY hKey, _Reserved_ DWORD Reserved, _In_opt_ DWORD dwOptions, _In_ REGSAM samDesired, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult, _Out_opt_ LPDWORD lpdwDisposition)                                                                               |
| tsuisuiuittttui | LONG | RegCreateKeyTransacted(_In_ HKEY hKey, _Reserved_ DWORD Reserved, _In_opt_ DWORD dwOptions, _In_ REGSAM samDesired, _In_opt_ const LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult, _Out_opt_ LPDWORD lpdwDisposition, _In_ HANDLE hTransaction, _Reserved_ PVOID pExtendedParameters)  |
| tauiauiuittttui | LONG | RegCreateKeyTransactedA(_In_ HKEY hKey, _Reserved_ DWORD Reserved, _In_opt_ DWORD dwOptions, _In_ REGSAM samDesired, _In_opt_ const LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult, _Out_opt_ LPDWORD lpdwDisposition, _In_ HANDLE hTransaction, _Reserved_ PVOID pExtendedParameters) |
| twuiwuiuittttui | LONG | RegCreateKeyTransactedW(_In_ HKEY hKey, _Reserved_ DWORD Reserved, _In_opt_ DWORD dwOptions, _In_ REGSAM samDesired, _In_opt_ const LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult, _Out_opt_ LPDWORD lpdwDisposition, _In_ HANDLE hTransaction, _Reserved_ PVOID pExtendedParameters) |
| twtui           | LONG | RegCreateKeyW(_In_ HKEY hKey, _Reserved_ DWORD Reserved, _In_opt_ DWORD dwOptions, _In_ REGSAM samDesired, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _Out_ PHKEY phkResult)                                                                                                                    |
| tsui            | LONG | RegDeleteKey(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey)                                                                                                                                                                                                                                                       |
| taui            | LONG | RegDeleteKeyA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey)                                                                                                                                                                                                                                                      |
| tsuiiui         | LONG | RegDeleteKeyEx(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_ REGSAM samDesired, _Reserved_)                                                                                                                                                                                                                 |
| tauuiui         | LONG | RegDeleteKeyExA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_ REGSAM samDesired, _Reserved_)                                                                                                                                                                                                                |
| twuiiui         | LONG | RegDeleteKeyExW(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_ REGSAM samDesired, _Reserved_)                                                                                                                                                                                                                |
| tsuiuittui      | LONG | RegDeleteKeyTransacted(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_ REGSAM samDesired, _Reserved_)                                                                                                                                                                                                         |

|             |      |                                                                                                                                                                                          |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |      | pExtendedParameter)                                                                                                                                                                      |
| tauiuittui  | LONG | RegDeleteKeyTransactedA(_In_ HKEY lpSubKey, _In_ REGSAM samDesired, _Reserved, _In_ HANDLE hTransaction, pExtendedParameter)                                                             |
| twuiuittui  | LONG | RegDeleteKeyTransactedW(_In_ HKEY lpSubKey, _In_ REGSAM samDesired, _Reserved, _In_ HANDLE hTransaction, pExtendedParameter)                                                             |
| tssui       | LONG | RegDeleteKeyValue(_In_ HKEY hKey, _In_ lpSubKey, _In_opt_ LPCTSTR lpValueName)                                                                                                           |
| taai        | LONG | RegDeleteKeyValueA(_In_ HKEY hKey, _In_ lpSubKey, _In_opt_ LPCSTR lpValueName)                                                                                                           |
| twwai       | LONG | RegDeleteKeyValueW(_In_ HKEY hKey, _In_ lpSubKey, _In_opt_ LPCWSTR lpValueName)                                                                                                          |
| twui        | LONG | RegDeleteKeyW(_In_ HKEY hKey, _In_ lpSubKey)                                                                                                                                             |
| tsui        | LONG | RegDeleteTree(_In_ HKEY hKey, _In_ lpSubKey)                                                                                                                                             |
| taui        | LONG | RegDeleteTreeA(_In_ HKEY hKey, _In_ lpSubKey)                                                                                                                                            |
| twui        | LONG | RegDeleteTreeW(_In_ HKEY hKey, _In_ lpSubKey)                                                                                                                                            |
| tsui        | LONG | RegDeleteValue(_In_ HKEY hKey, _In_ lpValueName)                                                                                                                                         |
| taui        | LONG | RegDeleteValueA(_In_ HKEY hKey, _In_ lpValueName)                                                                                                                                        |
| twui        | LONG | RegDeleteValueW(_In_ HKEY hKey, _In_ lpValueName)                                                                                                                                        |
| ui          | LONG | RegDisablePredefinedCache(void)                                                                                                                                                          |
| ui          | LONG | RegDisablePredefinedCacheEx(void)                                                                                                                                                        |
| tui         | LONG | RegDisableReflectionKey(_In_ HKEY hKey)                                                                                                                                                  |
| tui         | LONG | RegEnableReflectionKey(_In_ HKEY hKey)                                                                                                                                                   |
| tuisuiui    | LONG | RegEnumKey(_In_ HKEY hKey, _In_ DWORD dwIndex, _Out_ LPTSTR lpName, _In_ DWORD cchName)                                                                                                  |
| tuiuiui     | LONG | RegEnumKeyA(_In_ HKEY hKey, _In_ DWORD dwIndex, _Out_ LPSTR lpName, _In_ DWORD cchName)                                                                                                  |
| tuiiststtui | LONG | RegEnumKeyEx(_In_ HKEY hKey, _In_ DWORD dwIndex, _Out_ LPTSTR lpName, _Inout_ LPDWORD lpReserved, _In_ DWORD dwOptions, _Inout_opt_ LPDWORD lpClass, _Out_ LPFILETIME lpftLastWriteTime) |
|             |      | RegEnumKeyExA(_In_ HKEY hKey, _In_                                                                                                                                                       |

|             |        |                                                                                                                                                                                     |
|-------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiattattui | LONG   | _Out_ LPSTR lpName, _Inout_ LPDWO<br>_Reserved_ LPDWORD lpReserved, _In<br>_Inout_opt_ LPDWORD lpcClass, _Out<br>lpftLastWriteTime)                                                 |
| tuiwttwtui  | LONG   | RegEnumKeyExW(_In_ HKEY hKey, _<br>_Out_ LPWSTR lpName, _Inout_ LPDV<br>_Reserved_ LPDWORD lpReserved, _In<br>_Inout_opt_ LPDWORD lpcClass, _Out<br>lpftLastWriteTime)              |
| tuiwuiui    | LONG   | RegEnumKeyW(_In_ HKEY hKey, _In_<br>_Out_ LPWSTR lpName, _In_ DWORD                                                                                                                 |
| tuiistttui  | LONG   | RegEnumValue(_In_ HKEY hKey, _In_<br>_Out_ LPTSTR lpValueName, _Inout_ I<br>lpchValueName, _Reserved_ LPDWOF<br>_Out_opt_ LPDWORD lpType, _Out_op<br>_Inout_opt_ LPDWORD lpcbData)  |
| tuiattttui  | LONG   | RegEnumValueA(_In_ HKEY hKey, _In_<br>_Out_ LPSTR lpValueName, _Inout_ LP<br>lpchValueName, _Reserved_ LPDWOF<br>_Out_opt_ LPDWORD lpType, _Out_op<br>_Inout_opt_ LPDWORD lpcbData) |
| tuiwttttui  | LONG   | RegEnumValueW(_In_ HKEY hKey, _In_<br>_Out_ LPWSTR lpValueName, _Inout_<br>lpchValueName, _Reserved_ LPDWOF<br>_Out_opt_ LPDWORD lpType, _Out_op<br>_Inout_opt_ LPDWORD lpcbData)   |
| tui         | LONG   | RegFlushKey(_In_ HKEY hKey)                                                                                                                                                         |
| tuittui     | LONG   | RegGetKeySecurity(_In_ HKEY hKey, _<br>SECURITY_INFORMATION SecurityI<br>PSECURITY_DESCRIPTOR pSecurityD<br>LPDWORD lpcbSecurityDescriptor)                                         |
| tssuittui   | LONG   | RegGetValue(_In_ HKEY hkey, _In_opt<br>_In_opt_ LPCTSTR lpValue, _In_opt_ D<br>_Out_opt_ LPDWORD pdwType, _Out_<br>_Inout_opt_ LPDWORD pcbData)                                     |
| taauittui   | LONG   | RegGetValueA(_In_ HKEY hkey, _In_o<br>_In_opt_ LPCSTR lpValue, _In_opt_ DV<br>_Out_opt_ LPDWORD pdwType, _Out_<br>_Inout_opt_ LPDWORD pcbData)                                      |
| twwuittui   | LONG   | RegGetValueW(_In_ HKEY hkey, _In_o<br>lpSubKey, _In_opt_ LPCWSTR lpValue<br>dwFlags, _Out_opt_ LPDWORD pdwTy<br>pvData, _Inout_opt_ LPDWORD pcbDa                                   |
| sst         | HANDLE | RegisterEventSource(_In_ LPCTSTR lp<br>LPCTSTR lpSourceName)                                                                                                                        |
| aat         | HANDLE | RegisterEventSourceA(_In_ LPCSTR lp                                                                                                                                                 |

|             |                       |                                                                                                                                                       |
|-------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |                       | LPCSTR lpSourceName)                                                                                                                                  |
| wwt         | HANDLE                | RegisterEventSourceW(_In_ LPCWSTR<br>_In_ LPCWSTR lpSourceName)                                                                                       |
| stt         | SERVICE_STATUS_HANDLE | RegisterServiceCtrlHandler(_In_ LPCTS<br>_In_ LPHANDLER_FUNCTION lpHan                                                                                |
| att         | SERVICE_STATUS_HANDLE | RegisterServiceCtrlHandlerA(_In_ LPCS<br>_In_ LPHANDLER_FUNCTION lpHan                                                                                |
| sttt        | SERVICE_STATUS_HANDLE | RegisterServiceCtrlHandlerEx(_In_ LPC<br>_In_ LPHANDLER_FUNCTION_EX lp<br>LPVOID lpContext)                                                           |
| attt        | SERVICE_STATUS_HANDLE | RegisterServiceCtrlHandlerExA(_In_ LP<br>_In_ LPHANDLER_FUNCTION_EX lp<br>LPVOID lpContext)                                                           |
| wttt        | SERVICE_STATUS_HANDLE | RegisterServiceCtrlHandlerExW(_In_ L<br>lpServiceName, _In_ LPHANDLER_FU<br>lpHandlerProc, _In_opt_ LPVOID lpCor                                      |
| wtt         | SERVICE_STATUS_HANDLE | RegisterServiceCtrlHandlerW(_In_ LPC<br>_In_ LPHANDLER_FUNCTION lpHan                                                                                 |
| titi        | VOID                  | RegisterWaitChainCOMCallback(_In_ P<br>CallStateCallback, _In_ PCOGETACTIV<br>ActivationStateCallback)                                                |
| stuiuiuiui  | LONG                  | RegLoadAppKey(_In_ LPCTSTR lpFile<br>phkResult, _In_ REGSAM samDesired,<br>dwOptions, _Reserved_ DWORD Reser                                          |
| atuiuiuiui  | LONG                  | RegLoadAppKeyA(_In_ LPCSTR lpFile<br>phkResult, _In_ REGSAM samDesired,<br>dwOptions, _Reserved_ DWORD Reser                                          |
| wtuiuiuiui  | LONG                  | RegLoadAppKeyW(_In_ LPCWSTR lpF<br>phkResult, _In_ REGSAM samDesired,<br>dwOptions, _Reserved_ DWORD Reser                                            |
| tssui       | LONG                  | RegLoadKey(_In_ HKEY hKey, _In_opt<br>_In_ LPCTSTR lpFile)                                                                                            |
| taaii       | LONG                  | RegLoadKeyA(_In_ HKEY hKey, _In_o<br>_In_ LPCSTR lpFile)                                                                                              |
| twwui       | LONG                  | RegLoadKeyW(_In_ HKEY hKey, _In_o<br>lpSubKey, _In_ LPCWSTR lpFile)                                                                                   |
| tssuituisui | LONG                  | RegLoadMUIString(_In_ HKEY hKey, _<br>pszValue, _Out_opt_ LPTSTR pszOutBu<br>cbOutBuf, _Out_opt_ LPDWORD pcbD<br>Flags, _In_opt_ LPCTSTR pszDirectory |
| taaituiaui  | LONG                  | RegLoadMUIStringA(_In_ HKEY hKey<br>pszValue, _Out_opt_ LPSTR pszOutBuf<br>cbOutBuf, _Out_opt_ LPDWORD pcbD<br>Flags, _In_opt_ LPCSTR pszDirectory)   |

|             |      |                                                                                                                                                                                                     |
|-------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twwuituiwui | LONG | RegLoadMUIStringW(_In_ HKEY hKey, _In_ LPWSTR pszValue, _Out_opt_ LPWSTR pszOutBuf, _Out_opt_ LPDWORD pcbOutBuf, _Out_opt_ LPDWORD pcbDirFlags, _In_opt_ LPCWSTR pszDirectory)                      |
| tiuitui     | LONG | RegNotifyChangeKeyValue(_In_ HKEY hKey, _In_ DWORD dwNotifyFilter, _In_ HANDLE hEvent, _In_ BOOL fAsynchronous)                                                                                     |
| uitui       | LONG | RegOpenCurrentUser(_In_ REGSAM samDesired, _Out_ PHKEY phkResult)                                                                                                                                   |
| tstui       | LONG | RegOpenKey(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Out_ PHKEY phkResult)                                                                                                                            |
| tatui       | LONG | RegOpenKeyA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Out_ PHKEY phkResult)                                                                                                                           |
| tsuiuitui   | LONG | RegOpenKeyEx(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Reserved_ DWORD ulOptions, _In_ REGSAM samDesired, _Out_ PHKEY phkResult)                                                                      |
| tauiuitui   | LONG | RegOpenKeyExA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Reserved_ DWORD ulOptions, _In_ REGSAM samDesired, _Out_ PHKEY phkResult)                                                                     |
| twuiuitui   | LONG | RegOpenKeyExW(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Reserved_ DWORD ulOptions, _In_ REGSAM samDesired, _Out_ PHKEY phkResult)                                                                     |
| tsuiuittui  | LONG | RegOpenKeyTransacted(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_ DWORD ulOptions, _In_ HANDLE hTransaction, _Out_ PHKEY phkResult, _In_ HANDLE hTransaction, _Reserved_ PVOID pExtendedParameters)  |
| tauiuittui  | LONG | RegOpenKeyTransactedA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_ DWORD ulOptions, _In_ HANDLE hTransaction, _Out_ PHKEY phkResult, _In_ HANDLE hTransaction, _Reserved_ PVOID pExtendedParameters) |
| twuiuittui  | LONG | RegOpenKeyTransactedW(_In_ HKEY hKey, _In_ LPCWSTR lpSubKey, _In_ DWORD ulOptions, _In_ HANDLE hTransaction, _Out_ PHKEY phkResult, _In_ HANDLE hTransaction, _Reserved_ PVOID pExtendedParameters) |
| twtui       | LONG | RegOpenKeyW(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Out_ PHKEY phkResult)                                                                                                                           |
| tuiuitui    | LONG | RegOpenUserClassesRoot(_In_ HANDLE hUser, _In_ DWORD dwOptions, _In_ REGSAM samDesired, _Out_ PHKEY phkResult)                                                                                      |
| ttui        | LONG | RegOverridePredefKey(_In_ HKEY hKey, _In_ HKEY hNewHKey)                                                                                                                                            |
|             |      | RegQueryInfoKey(_In_ HKEY hKey, _Out_opt_ LPDWORD lpClass, _Inout_opt_ LPDWORD lpClassSize, _Out_opt_ LPDWORD lpReserved, _Out_opt_ LPDWORD lpMaxSubKeyIndex)                                       |

|                    |      |                                                                                                                                                                                                                                                                                                                                             |
|--------------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsTTTTTTTTtTui     | LONG | LPDWORD lpcMaxClassLen, _Out_opt_ LPDWORD lpcMaxValueNameLen, _Out_opt_ LPDWORD lpcMaxValueLen, _Out_opt_ PFID lpcbSecurityDescriptor, _Out_opt_ PFID lpftLastWriteTime)                                                                                                                                                                    |
| tattTTTTTTTTtTui   | LONG | RegQueryInfoKeyA(_In_ HKEY hKey, _In_ LPCTSTR lpClass, _Inout_opt_ LPDWORD lpcClassLen, _Out_opt_ LPDWORD lpReserved, _Out_opt_ LPDWORD lpcMaxSubKeyLen, _Out_opt_ LPDWORD lpcMaxClassLen, _Out_opt_ LPDWORD lpcMaxValueNameLen, _Out_opt_ LPDWORD lpcMaxValueLen, _Out_opt_ PFID lpcbSecurityDescriptor, _Out_opt_ PFID lpftLastWriteTime) |
| twTTTTTTTTTTTTtTui | LONG | RegQueryInfoKeyW(_In_ HKEY hKey, _In_ LPCTSTR lpClass, _Inout_opt_ LPDWORD lpcClassLen, _Out_opt_ LPDWORD lpReserved, _Out_opt_ LPDWORD lpcMaxSubKeyLen, _Out_opt_ LPDWORD lpcMaxClassLen, _Out_opt_ LPDWORD lpcMaxValueNameLen, _Out_opt_ LPDWORD lpcMaxValueLen, _Out_opt_ PFID lpcbSecurityDescriptor, _Out_opt_ PFID lpftLastWriteTime) |
| ttuistui           | LONG | RegQueryMultipleValues(_In_ HKEY hKey, _In_ LPCTSTR val_list, _In_ DWORD num_vals, _Out_ LPVOID lpValueBuf, _Inout_opt_ LPDWORD lpdwDisposition)                                                                                                                                                                                            |
| ttuiatui           | LONG | RegQueryMultipleValuesA(_In_ HKEY hKey, _In_ LPCTSTR val_list, _In_ DWORD num_vals, _Out_ LPVOID lpValueBuf, _Inout_opt_ LPDWORD lpdwDisposition)                                                                                                                                                                                           |
| ttuiwtui           | LONG | RegQueryMultipleValuesW(_In_ HKEY hKey, _In_ LPCTSTR val_list, _In_ DWORD num_vals, _Out_ LPVOID lpValueBuf, _Inout_opt_ LPDWORD lpdwDisposition)                                                                                                                                                                                           |
| ttui               | LONG | RegQueryReflectionKey(_In_ HKEY hKey, _Out_ BOOL *bIsReflectionDisabled)                                                                                                                                                                                                                                                                    |
| tsstui             | LONG | RegQueryValue(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Out_opt_ LPTSTR lpValue, _Out_opt_ DWORD lpcbValue)                                                                                                                                                                                                                                   |
| taatui             | LONG | RegQueryValueA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _Out_opt_ LPSTR lpValue, _Out_opt_ DWORD lpcbValue)                                                                                                                                                                                                                                   |
| tsTTTTtTui         | LONG | RegQueryValueEx(_In_ HKEY hKey, _In_ LPCTSTR lpValueName, _Reserved_ LPDWORD lpReserved, _In_ LPDWORD lpType, _Out_opt_ LPBYTE lpData, _Out_opt_ LPDWORD lpcbData)                                                                                                                                                                          |
| tattTTTTtTui       | LONG | RegQueryValueExA(_In_ HKEY hKey, _In_ LPCTSTR lpValueName, _Reserved_ LPDWORD lpReserved, _In_ LPDWORD lpType, _Out_opt_ LPBYTE lpData, _Out_opt_ LPDWORD lpcbData)                                                                                                                                                                         |

|            |      |                                                                                                                                                       |
|------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |      | LPDWORD lpcbData)                                                                                                                                     |
| twtttui    | LONG | RegQueryValueExW(_In_ HKEY hKey, lpValueName, _Reserved_ LPDWORD lpType, _Out_opt_ LPBYTE lpData, LPDWORD lpcbData)                                   |
| twwtui     | LONG | RegQueryValueW(_In_ HKEY hKey, _In_ lpSubKey, _Out_opt_ LPWSTR lpValue, lpcbValue)                                                                    |
| tsssui     | LONG | RegReplaceKey(_In_ HKEY hKey, _In_ lpSubKey, _In_ LPCTSTR lpNewFile, _In_ lpOldFile)                                                                  |
| taaaui     | LONG | RegReplaceKeyA(_In_ HKEY hKey, _In_ lpSubKey, _In_ LPCSTR lpNewFile, _In_ lpOldFile)                                                                  |
| twwwui     | LONG | RegReplaceKeyW(_In_ HKEY hKey, _In_ lpSubKey, _In_ LPCWSTR lpNewFile, _In_ lpOldFile)                                                                 |
| tsuiui     | LONG | RegRestoreKey(_In_ HKEY hKey, _In_ DWORD dwFlags)                                                                                                     |
| tauiui     | LONG | RegRestoreKeyA(_In_ HKEY hKey, _In_ DWORD dwFlags)                                                                                                    |
| twuiui     | LONG | RegRestoreKeyW(_In_ HKEY hKey, _In_ _In_ DWORD dwFlags)                                                                                               |
| tstui      | LONG | RegSaveKey(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_ LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                      |
| tatui      | LONG | RegSaveKeyA(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_ LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                     |
| tstuiui    | LONG | RegSaveKeyEx(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD Flags)                              |
| tatuiui    | LONG | RegSaveKeyExA(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD Flags)                             |
| twtuiui    | LONG | RegSaveKeyExW(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD Flags)                             |
| twtui      | LONG | RegSaveKeyW(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                 |
| tuitui     | LONG | RegSetKeySecurity(_In_ HKEY hKey, _In_ LPCTSTR lpPath, _In_ SECURITY_INFORMATION SecurityInformation, _In_ LPSECURITY_DESCRIPTOR pSecurityDescriptor) |
| tssuituiui | LONG | RegSetKeyValue(_In_ HKEY hKey, _In_ lpSubKey, _In_opt_ LPCTSTR lpValueName, _In_ DWORD dwType, _In_opt_ LPCVOID lpData, _In_ LPDWORD lpcbData)        |
|            |      |                                                                                                                                                       |

|                |       |                                                                                                                                                                                                     |
|----------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| taauituiui     | LONG  | RegSetValueA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_opt_ LPCWSTR lpValueName, _In_ DWORD dwType, _In_opt_ LPCVOID lpData, _In_ DWORD dwFlags)                                                   |
| twwwituiui     | LONG  | RegSetValueW(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey, _In_opt_ LPCWSTR lpValueName, _In_ DWORD dwType, _In_opt_ LPCVOID lpData, _In_ DWORD dwFlags)                                                   |
| tsuisuiui      | LONG  | RegSetValue(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwType, _In_ LPCTSTR lpValueName, _In_ DWORD dwType, _In_ LPCTSTR lpData, _In_ DWORD dwFlags)                                     |
| tauiuiui       | LONG  | RegSetValueA(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwType, _In_ LPCTSTR lpValueName, _In_ DWORD dwType, _In_ LPCTSTR lpData, _In_ DWORD dwFlags)                                    |
| tsuiuituiui    | LONG  | RegSetValueEx(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwType, _In_ LPCTSTR lpValueName, _Reserved_ DWORD dwType, _In_ const BYTE *lpData, _In_ DWORD dwFlags)                         |
| tauiuituiui    | LONG  | RegSetValueExA(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwType, _In_ LPCTSTR lpValueName, _Reserved_ DWORD dwType, _In_ const BYTE *lpData, _In_ DWORD dwFlags)                        |
| twuiuituiui    | LONG  | RegSetValueExW(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwType, _In_ LPCTSTR lpValueName, _Reserved_ DWORD dwType, _In_ const BYTE *lpData, _In_ DWORD dwFlags)                        |
| twuiwuiui      | LONG  | RegSetValueW(_In_ HKEY hKey, _In_opt_ LPCTSTR lpSubKey, _In_ DWORD dwType, _In_ LPCTSTR lpValueName, _Reserved_ DWORD dwType, _In_ const BYTE *lpData, _In_ DWORD dwFlags)                          |
| tsui           | LONG  | RegUnLoadKey(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey)                                                                                                                                                 |
| taui           | LONG  | RegUnLoadKeyA(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey)                                                                                                                                                |
| twui           | LONG  | RegUnLoadKeyW(_In_ HKEY hKey, _In_ LPCTSTR lpSubKey)                                                                                                                                                |
| wtui           | DWORD | RemoveUsersFromEncryptedFile(_In_ LPCTSTR lpFile, _In_ PENCIPHERING_CERTIFICATE_HANDLE hCertificate)                                                                                                |
| tuhuhituhuisti | BOOL  | ReportEvent(_In_ HANDLE hEventLog, _In_ WORD wCategory, _In_ DWORD dwLevel, _In_ LPCTSTR lpUserSid, _In_ WORD wNumStrings, _In_ DWORD dwDataSize, _In_ LPCTSTR *lpStrings, _In_ LPCTSTR lpRawData)  |
| tuhuhituhuiati | BOOL  | ReportEventA(_In_ HANDLE hEventLog, _In_ WORD wCategory, _In_ DWORD dwLevel, _In_ LPCTSTR lpUserSid, _In_ WORD wNumStrings, _In_ DWORD dwDataSize, _In_ LPCTSTR *lpStrings, _In_ LPCTSTR lpRawData) |
| tuhuhituhuiwti | BOOL  | ReportEventW(_In_ HANDLE hEventLog, _In_ WORD wCategory, _In_ DWORD dwLevel, _In_ LPCTSTR lpUserSid, _In_ WORD wNumStrings, _In_ DWORD dwDataSize, _In_ LPCTSTR *lpStrings, _In_ LPCTSTR lpRawData) |

|           |       |                                                                                                                                                               |
|-----------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |       | dwDataSize, _In_ LPCWSTR *lpStrings<br>lpRawData)                                                                                                             |
| i         | BOOL  | RevertToSelf(void)                                                                                                                                            |
| ti        | BOOL  | SaferCloseLevel(_In_ SAFER_LEVEL_...<br>hLevelHandle)                                                                                                         |
| tttuiti   | BOOL  | SaferComputeTokenFromLevel(_In_...<br>SAFER_LEVEL_HANDLE LevelHandle<br>InAccessToken, _Out_ PHANDLE OutA...<br>DWORD dwFlags, _Inout_opt_ LPVOID             |
| uiuiuitti | BOOL  | SaferCreateLevel(_In_ DWORD dwScop...<br>dwLevelId, _In_ DWORD OpenFlags, _...<br>SAFER_LEVEL_HANDLE *pLevelHan...<br>LPVOID lpReserved)                      |
| tuituiti  | BOOL  | SaferGetLevelInformation(_In_ SAFER...<br>LevelHandle, _In_ SAFER_OBJECT_IN...<br>dwInfoType, _Out_opt_ LPVOID lpQue...<br>dwInBufferSize, _Out_ LPDWORD lpdv |
| uiuiuitti | BOOL  | SaferGetPolicyInformation(_In_ DWOR...<br>SAFER_POLICY_INFO_CLASS SaferP...<br>DWORD InfoBufferSize, _Out_ PVOID...<br>PDWORD InfoBufferRetSize, _Reserve     |
| uittti    | BOOL  | SaferIdentifyLevel(_In_ DWORD dwNu...<br>PSAFER_CODE_PROPERTIES pCodeP...<br>SAFER_LEVEL_HANDLE *pLevelHan...<br>LPVOID lpReserved)                           |
| wuci      | BOOL  | SaferIsExecutableFileType(_In_ LPCW...<br>BOOLEAN bFromShellExecute)                                                                                          |
| twti      | BOOL  | SaferRecordEventLogEntry(_In_ SAFE...<br>hLevel, _In_ LPCWSTR szTargetPath, _...<br>lpReserved)                                                               |
| tuituii   | BOOL  | SaferSetLevelInformation(_In_ SAFER...<br>LevelHandle, _In_ SAFER_OBJECT_IN...<br>dwInfoType, _In_ LPVOID lpQueryBuf...<br>dwInBufferSize)                    |
| uiuiuitti | BOOL  | SaferSetPolicyInformation(_In_ DWOR...<br>SAFER_POLICY_INFO_CLASS SaferP...<br>DWORD InfoBufferSize, _In_ PVOID I...<br>LPVOID lpReserved)                    |
| ttuiii    | BOOL  | SetAclInformation(_Inout_ PACL pAcl...<br>pAclInformation, _In_ DWORD nAclInf...<br>ACL_INFORMATION_CLASS dwAclI                                              |
| uitttui   | DWORD | SetEntriesInAcl(_In_ ULONG cCountO...<br>_In_opt_ PEXPLICIT_ACCESS pListO...<br>_In_opt_ PACL OldAcl, _Out_ PACL *N                                           |
|           |       | SetEntriesInAclA(_In_ ULONG cCount                                                                                                                            |

|            |       |                                                                                                                                                                                                                                                                                      |
|------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uitttui    | DWORD | <code>_In_opt_ PEXPLICIT_ACCESS pListOfEntries, _In_opt_ PACL OldAcl, _Out_ PACL *NewAcl)</code>                                                                                                                                                                                     |
| uitttui    | DWORD | <code>SetEntriesInAclW(_In_ ULONG cCount, _In_opt_ PEXPLICIT_ACCESS pListOfEntries, _In_opt_ PACL OldAcl, _Out_ PACL *NewAcl)</code>                                                                                                                                                 |
| suiti      | BOOL  | <code>SetFileSecurity(_In_ LPCTSTR lpFileName, SECURITY_INFORMATION SecurityInformation, PSECURITY_DESCRIPTOR pSecurityDescriptor)</code>                                                                                                                                            |
| auiti      | BOOL  | <code>SetFileSecurityA(_In_ LPCSTR lpFileName, SECURITY_INFORMATION SecurityInformation, PSECURITY_DESCRIPTOR pSecurityDescriptor)</code>                                                                                                                                            |
| wuiti      | BOOL  | <code>SetFileSecurityW(_In_ LPCWSTR lpFileName, SECURITY_INFORMATION SecurityInformation, PSECURITY_DESCRIPTOR pSecurityDescriptor)</code>                                                                                                                                           |
| tuiti      | BOOL  | <code>SetKernelObjectSecurity(_In_ HANDLE hObject, SECURITY_INFORMATION SecurityInformation, PSECURITY_DESCRIPTOR SecurityDescriptor)</code>                                                                                                                                         |
| suiuitttui | DWORD | <code>SetNamedSecurityInfo(_In_ LPTSTR pObjectName, SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation, psidOwner, _In_opt_ PSID psidGroup, _In_opt_ PACL pSacl)</code>                                                                                        |
| auiuitttui | DWORD | <code>SetNamedSecurityInfoA(_In_ LPSTR pObjectName, SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation, psidOwner, _In_opt_ PSID psidGroup, _In_opt_ PACL pSacl)</code>                                                                                        |
| wuiuitttui | DWORD | <code>SetNamedSecurityInfoW(_In_ LPWSTR pObjectName, SE_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation, psidOwner, _In_opt_ PSID psidGroup, _In_opt_ PACL pSacl)</code>                                                                                       |
| uitttti    | BOOL  | <code>SetPrivateObjectSecurity(_In_ SECURITY_INFORMATION SecurityInformation, _In_ PSECURITY_DESCRIPTOR ModificationDescriptor, _Inout_ PSECURITY_DESCRIPTOR *ObjectsSecurityDescriptor, _In_ PGENERIC_MAPPING GenericMapping, _In_opt_ HANDLE Token)</code>                         |
| uittuitti  | BOOL  | <code>SetPrivateObjectSecurityEx(_In_ SECURITY_INFORMATION SecurityInformation, _In_ PSECURITY_DESCRIPTOR ModificationDescriptor, _Inout_ PSECURITY_DESCRIPTOR *ObjectsSecurityDescriptor, _In_ ULONG_PTR GenericMapping, _In_ PGENERIC_MAPPING GenericMapping, HANDLE Token)</code> |
| uiti       | VOID  | <code>SetSecurityAccessMask(_In_ SECURITY_INFORMATION SecurityInformation, _Out_ LPDWORD pAccessMask)</code>                                                                                                                                                                         |
|            |       | <code>SetSecurityDescriptorControl(_In_ PSECURITY_DESCRIPTOR pSecurityDescriptor, SECURITY_INFORMATION SecurityInformation, SECURITY_DESCRIPTOR_CONTROL Control)</code>                                                                                                              |

|            |       |                                                                                                                                                                                             |
|------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuhuhi     | BOOL  | pSecurityDescriptor, _In_<br>SECURITY_DESCRIPTOR_CONTROL<br>_In_ SECURITY_DESCRIPTOR_CONTROL                                                                                                |
| titii      | BOOL  | SetSecurityDescriptorDacl(_Inout_ PSECURITY_DESCRIPTOR pSecurityDescriptor, _In_ BOOL bDaclPresent, _In_ PSECURITY_DESCRIPTOR pDacl, _In_ BOOL bDaclDefaulted)                              |
| tiii       | BOOL  | SetSecurityDescriptorGroup(_Inout_ PSECURITY_DESCRIPTOR pSecurityDescriptor, _In_ PSID pGroup, _In_ BOOL bGroupDefaulted)                                                                   |
| tiii       | BOOL  | SetSecurityDescriptorOwner(_Inout_ PSECURITY_DESCRIPTOR pSecurityDescriptor, _In_ PSID pOwner, _In_ BOOL bOwnerDefaulted)                                                                   |
| ttui       | DWORD | SetSecurityDescriptorRMControl(_Inout_ PSECURITY_DESCRIPTOR SecurityDescriptor, _In_ PCHAR RMControl)                                                                                       |
| titii      | BOOL  | SetSecurityDescriptorSacl(_Inout_ PSECURITY_DESCRIPTOR pSecurityDescriptor, _In_ BOOL bSaclPresent, _In_ PSECURITY_DESCRIPTOR pSacl, _In_ BOOL bSaclDefaulted)                              |
| tuiuitttui | DWORD | SetSecurityInfo(_In_ HANDLE handle, _In_ SECURITY_OBJECT_TYPE ObjectType, _In_ SECURITY_INFORMATION SecurityInformation, _In_ PSID psidOwner, _In_opt_ PSID psidGroup, _In_opt_ PACL pSacl) |
| tuiiii     | BOOL  | SetServiceBits(_In_ SERVICE_STATUS_HANDLE hServiceStatus, _In_ DWORD dwServiceBits, _In_ BOOL bSetBitsOn, _In_ BOOL bUpdateImmediate)                                                       |
| tuiti      | BOOL  | SetServiceObjectSecurity(_In_ SC_HANDLE hService, _In_ SECURITY_INFORMATION dwSecurityInformation, _In_ PSECURITY_DESCRIPTOR lpSecurityDescriptor)                                          |
| tii        | BOOL  | SetServiceStatus(_In_ SERVICE_STATUS_HANDLE hServiceStatus, _In_ LPSERVICE_STATUS lpServiceStatus)                                                                                          |
| tii        | BOOL  | SetThreadToken(_In_opt_ PHANDLE Token, _In_ HANDLE Token)                                                                                                                                   |
| tuituii    | BOOL  | SetTokenInformation(_In_ HANDLE Token, _In_ TOKEN_INFORMATION_CLASS TokenInformationClass, _In_ LPVOID TokenInformation, _In_ DWORD TokenInformationLength)                                 |
| tui        | DWORD | SetUserFileEncryptionKey(_In_ PENCIPHERING_CERTIFICATE pEncryptionCertificate, _In_ DWORD dwFileEncryptionKey)                                                                              |
| tuisi      | BOOL  | StartService(_In_ SC_HANDLE hService, _In_ DWORD dwNumServiceArgs, _In_opt_ LPCTSTR lpServiceName)                                                                                          |
| tuiiai     | BOOL  | StartServiceA(_In_ SC_HANDLE hService, _In_ DWORD dwNumServiceArgs, _In_opt_ LPCSTR lpServiceName)                                                                                          |
| ti         | BOOL  | StartServiceCtrlDispatcher(_In_ const SERVICE_TABLE_ENTRY *lpServiceTable)                                                                                                                  |

|                 |       |                                                                                                                                                                                                                                                                         |
|-----------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti              | BOOL  | StartServiceCtrlDispatcherA(_In_ const SERVICE_TABLE_ENTRY *lpServiceT                                                                                                                                                                                                  |
| ti              | BOOL  | StartServiceCtrlDispatcherW(_In_ const SERVICE_TABLE_ENTRY *lpServiceT                                                                                                                                                                                                  |
| tuiwi           | BOOL  | StartServiceW(_In_ SC_HANDLE hSer<br>dwNumServiceArgs, _In_opt_ LPCWST<br>*lpServiceArgVectors)                                                                                                                                                                         |
| ttui            | ULONG | TraceEvent(_In_ TRACEHANDLE Sess<br>PEVENT_TRACE_HEADER EventTrac                                                                                                                                                                                                       |
| tttui           | ULONG | TraceEventInstance(_In_ TRACEHAND<br>PEVENT_INSTANCE_HEADER Event<br>PEVENT_INSTANCE_INFO pInstInfo,<br>PEVENT_INSTANCE_INFO pParentIns                                                                                                                                 |
| tuituhtui       | ULONG | TraceMessage(_In_ TRACEHANDLE S<br>ULONG MessageFlags, _In_ LPGUID M<br>USHORT MessageNumber, ...)                                                                                                                                                                      |
| tuituhtui       | ULONG | TraceMessageVa(_In_ TRACEHANDL<br>ULONG MessageFlags, _In_ LPGUID M<br>USHORT MessageNumber, _In_ va_list                                                                                                                                                               |
| suiuitttititui  | DWORD | TreeResetNamedSecurityInfo(_In_ LPT<br>SE_OBJECT_TYPE ObjectType, _In_<br>SECURITY_INFORMATION SecurityI<br>pOwner, _In_opt_ PSID pGroup, _In_op<br>_In_opt_ PACL pSacl, _In_ BOOL Keep<br>FN_PROGRESS fnProgress, _In_ PROC<br>ProgressInvokeSetting, _In_opt_ PVOID   |
| auuiuitttititui | DWORD | TreeResetNamedSecurityInfoA(_In_ LP<br>SE_OBJECT_TYPE ObjectType, _In_<br>SECURITY_INFORMATION SecurityI<br>pOwner, _In_opt_ PSID pGroup, _In_op<br>_In_opt_ PACL pSacl, _In_ BOOL Keep<br>FN_PROGRESS fnProgress, _In_ PROC<br>ProgressInvokeSetting, _In_opt_ PVOID   |
| wuiuitttititui  | DWORD | TreeResetNamedSecurityInfoW(_In_ LP<br>_In_ SE_OBJECT_TYPE ObjectType, _<br>SECURITY_INFORMATION SecurityI<br>pOwner, _In_opt_ PSID pGroup, _In_op<br>_In_opt_ PACL pSacl, _In_ BOOL Keep<br>FN_PROGRESS fnProgress, _In_ PROC<br>ProgressInvokeSetting, _In_opt_ PVOID |
| suiuitttuititui | DWORD | TreeSetNamedSecurityInfo(_In_ LPTST<br>SE_OBJECT_TYPE ObjectType, _In_<br>SECURITY_INFORMATION SecurityI<br>pOwner, _In_opt_ PSID pGroup, _In_op<br>_In_opt_ PACL pSacl, _In_ DWORD dv<br>FN_PROGRESS fnProgress, _In_ PROC<br>ProgressInvokeSetting, _In_opt_ PVOID    |

|                  |       |                                                                                                                                                                                                                                                                                       |
|------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aiiuitttuititui  | DWORD | <a href="#">TreeSetNamedSecurityInfoA</a> (_In_ LPST<br>SE_OBJECT_TYPE ObjectType, _In_<br>SECURITY_INFORMATION SecurityI<br>pOwner, _In_opt_ PSID pGroup, _In_op<br>_In_opt_ PACL pSacl, _In_ DWORD dv<br>FN_PROGRESS fnProgress, _In_ PROC<br>ProgressInvokeSetting, _In_opt_ PVOID |
| wuiiuitttuititui | DWORD | <a href="#">TreeSetNamedSecurityInfoW</a> (_In_ LPW<br>SE_OBJECT_TYPE ObjectType, _In_<br>SECURITY_INFORMATION SecurityI<br>pOwner, _In_opt_ PSID pGroup, _In_op<br>_In_opt_ PACL pSacl, _In_ DWORD dv<br>FN_PROGRESS fnProgress, _In_ PROC<br>ProgressInvokeSetting, _In_opt_ PVOID  |
| wuiui            | DWORD | <a href="#">UninstallApplication</a> (_In_ WCHAR *Pr<br>DWORD dwStatus)                                                                                                                                                                                                               |
| ti               | BOOL  | <a href="#">UnlockServiceDatabase</a> (_In_ SC_LOCK                                                                                                                                                                                                                                   |
| tui              | ULONG | <a href="#">UnregisterTraceGuids</a> (_In_ TRACEHAN<br>RegistrationHandle)                                                                                                                                                                                                            |
| tstui            | ULONG | <a href="#">UpdateTrace</a> (_In_ TRACEHANDLE Se<br>LPCTSTR SessionName, _Inout_<br>PEVENT_TRACE_PROPERTIES Prope                                                                                                                                                                     |
| tatui            | ULONG | <a href="#">UpdateTraceA</a> (_In_ TRACEHANDLE S<br>LPCSTR SessionName, _Inout_<br>PEVENT_TRACE_PROPERTIES Prope                                                                                                                                                                      |
| twtui            | ULONG | <a href="#">UpdateTraceW</a> (_In_ TRACEHANDLE S<br>LPCWSTR SessionName, _Inout_<br>PEVENT_TRACE_PROPERTIES Prope                                                                                                                                                                     |
| tttui            | DWORD | <a href="#">WriteEncryptedFileRaw</a> (_In_ PFE_IMP<br>pfImportCallback, _In_opt_ PVOID pvC<br>PVOID pvContext)                                                                                                                                                                       |

# Comctl32.dll

|                 |                |                                                                                                                                                                                                                                                |
|-----------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti              | BOOL           | <code>_TrackMouseEvent</code> ( <code>_Inout_ LPTRACKMOUSEEVENT lpEventTrack</code> )                                                                                                                                                          |
| tsi             | int            | <code>AddMRUStringW</code> ( <code>_In_ HANDLE hMRU, _In_ LPCTSTR szString</code> )                                                                                                                                                            |
| ttuitit         | HBITMAP        | <code>CreateMappedBitmap</code> ( <code>HINSTANCE hInstance, INT_PTR idBitmap, UINT wFlags, _In_ LPCOLORMAP lpColorMap, int iNumMaps</code> )                                                                                                  |
| ti              | int            | <code>CreateMRUListW</code> ( <code>_In_ LPMRUINFO lpmi</code> )                                                                                                                                                                               |
| tt              | HPROPSHEETPAGE | <code>CreatePropertySheetPage</code> ( <code>LPCPROPSHEETPAGE lppsp</code> )                                                                                                                                                                   |
| tt              | HPROPSHEETPAGE | <code>CreatePropertySheetPageA</code> ( <code>LPCPROPSHEETPAGE lppsp</code> )                                                                                                                                                                  |
| tt              | HPROPSHEETPAGE | <code>CreatePropertySheetPageW</code> ( <code>LPCPROPSHEETPAGE lppsp</code> )                                                                                                                                                                  |
| uistuit         | HWND           | <code>CreateStatusWindow</code> ( <code>LONG style, LPCTSTR lpszText, HWND hwndParent, UINT wID</code> )                                                                                                                                       |
| uiatuit         | HWND           | <code>CreateStatusWindowA</code> ( <code>LONG style, LPCSTR lpszText, HWND hwndParent, UINT wID</code> )                                                                                                                                       |
| uiwtuit         | HWND           | <code>CreateStatusWindowW</code> ( <code>LONG style, LPCWSTR lpszText, HWND hwndParent, UINT wID</code> )                                                                                                                                      |
| tuiuiitttiiiuit | HWND           | <code>CreateToolBarEx</code> ( <code>HWND hwnd, DWORD ws, UINT wID, int nBitmaps, HINSTANCE hBMInst, UINT_PTR wBMID, LPCTBBUTTON lpButtons, int iNumButtons, int dxButton, int dyButton, int dxBitmap, int dyBitmap, UINT uStructSize</code> ) |
| uiiiitttitiit   | HWND           | <code>CreateUpDownControl</code> ( <code>DWORD dwStyle, int x, int y, int cx, int cy, HWND hParent, int nID, HINSTANCE hInst, HWND hBuddy, int nUpper, int nLower, int nPos</code> )                                                           |
| tuittt          | LRESULT        | <code>DefSubclassProc</code> ( <code>_In_ HWND hWnd, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam</code> )                                                                                                                           |
| ti              | BOOL           | <code>DestroyPropertySheetPage</code> ( <code>HPROPSHEETPAGE hPSPPage</code> )                                                                                                                                                                 |
| ttt             | HDPA           | <code>DPA_Clone</code> ( <code>_In_ const HDPA hdpaSource, _Inout_opt_ HDPA hdpaNew</code> )                                                                                                                                                   |
| it              | HDPA           | <code>DPA_Create</code> ( <code>int cpGrow</code> )                                                                                                                                                                                            |
| itt             | HDPA           | <code>DPA_CreateEx</code> ( <code>_In_ int cpGrow, _In_opt_ HANDLE hheap</code> )                                                                                                                                                              |
| ti              | BOOL           | <code>DPA_DeleteAllPtrs</code> ( <code>HDPA pdpa</code> )                                                                                                                                                                                      |
| tit             | void*          | <code>DPA_DeletePtr</code> ( <code>HDPA pdpa, int index</code> )                                                                                                                                                                               |
| ti              | BOOL           | <code>DPA_Destroy</code> ( <code>HDPA pdpa</code> )                                                                                                                                                                                            |
| ttti            | VOID           | <code>DPA_DestroyCallback</code> ( <code>HDPA pdpa, PFNDPAENUMCALLBACK pfnCB, void *pData</code> )                                                                                                                                             |

|                |           |                                                                                                                                                        |
|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttti           | VOID      | DPA_EnumCallback (HDPA pdpa, PFNDPAENUMCALLBACK pfnCB, void *pData)                                                                                    |
| tit            | void*     | DPA_GetPtr (HDPA pdpa, int index)                                                                                                                      |
| tti            | int       | DPA_GetPtrIndex (_In_ HDPA hdpa, _In_ const void *pvoid)                                                                                               |
| tui6           | ULONGLONG | DPA_GetSize (_In_ HDPA pdpa)                                                                                                                           |
| tii            | BOOL      | DPA_Grow (_In_ HDPA hdpa, _In_ int cp)                                                                                                                 |
| titi           | int       | DPA_InsertPtr (HDPA pdpa, int index, void *p)                                                                                                          |
| ttti           | HRESULT   | DPA_LoadStream (_Out_ HDPA *ppdpa, _In_ PFNDPASTREAM pfn, _In_ IStream *pstm, _In_ void *pvInstData)                                                   |
| ttuitti        | BOOL      | DPA_Merge (_Inout_ HDPA hdpaDest, _In_ HDPA hdpaSrc, _In_ DWORD dwFlags, _In_ PFNDPACOMPARE pfnCompare, _In_ PFNDPAMERGE pfnMerge, _In_ LPARAM lParam) |
| ttti           | HRESULT   | DPA_SaveStream (_In_ HDPA pdpa, _In_ PFNDPASTREAM pfn, _In_ IStream *pstm, _In_ void *pvInstData)                                                      |
| ttittuii       | int       | DPA_Search (HDPA pdpa, void *pFind, int iStart, PFNDPACOMPARE pfnCmp, LPARAM lParam, UINT options)                                                     |
| titi           | BOOL      | DPA_SetPtr (HDPA pdpa, int index, void *p)                                                                                                             |
| ttti           | BOOL      | DPA_Sort (HDPA pdpa, PFNDPACOMPARE pfnCmp, LPARAM lParam)                                                                                              |
| tiii           | VOID      | DrawInset (HWND handParent, HWND hLB, int nItem)                                                                                                       |
| twuituiuiuiiii | int       | DrawShadowText (HDC hdc, LPCWSTR pszText, UINT cch, const RECT *pRect, DWORD dwFlags, COLORREF crText, COLORREF crShadow, int ixOffset, int iyOffset)  |
| ttsuii         | VOID      | DrawStatusText (HDC hdc, LPCRECT lprc, LPCTSTR pszText, UINT uFlags)                                                                                   |
| ttauii         | VOID      | DrawStatusTextA (HDC hdc, LPCRECT lprc, LPCSTR pszText, UINT uFlags)                                                                                   |
| ttwuii         | VOID      | DrawStatusTextW (HDC hdc, LPCRECT lprc, LPCWSTR pszText, UINT uFlags)                                                                                  |
| tt             | HDSA      | DSA_Clone (_In_ HDSA hdsa)                                                                                                                             |
| iit            | HDSA      | DSA_Create (_In_ int cbItem, _In_ int cbItemGrow)                                                                                                      |
| ti             | BOOL      | DSA_DeleteAllItems (_In_ HDSA hdsa)                                                                                                                    |
| tii            | BOOL      | DSA_DeleteItem (_In_ HDSA hdsa, _In_ int nPosition)                                                                                                    |
| ti             | BOOL      | DSA_Destroy (_In_ HDSA pdsa)                                                                                                                           |
| ttti           | VOID      | DSA_DestroyCallback (_In_ HDSA pdsa, _In_ PFNDSAENUMCALLBACK pfnCB, _In_ void *pData)                                                                  |
| ttti           | VOID      | DSA_EnumCallback (_In_ HDSA hdsa, _In_                                                                                                                 |

|         |           |                                                                                                                          |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------|
|         |           | PFNDAENUMCALLBACK *pfnCB, _In_ void *pData)                                                                              |
| titi    | BOOL      | DSA_GetItem(_In_ HDSA pdsa, _In_ int index, _Out_ void *pitem)                                                           |
| tit     | void*     | DSA_GetItemPtr(_In_ HDSA pdsa, _In_ int index)                                                                           |
| tui6    | ULONGLONG | DSA_GetSize(_In_ HDSA hdsa)                                                                                              |
| titi    | int       | DSA_InsertItem(_In_ HDSA pdsa, _In_ int index, _In_ void *pItem)                                                         |
| titi    | BOOL      | DSA_SetItem(_In_ HDSA hdsa, _In_ int index, _In_ void *pItem)                                                            |
| ttti    | BOOL      | DSA_Sort(_In_ HDSA pdsa, _In_ PFNDACOMPARE pfnCompare, _In_ LPARAM lParam)                                               |
| tituui  | int       | EnumMRUListW(_In_ HANDLE hMRU, _In_ int nItem, _Out_ void *lpData, _In_ UINT uLen)                                       |
| tiuui   | BOOL      | FlatSB_EnableScrollBar(HWND hwnd, int wSBflags, UINT wArrows)                                                            |
| titi    | BOOL      | FlatSB_GetScrollInfo(HWND hwnd, int fnBar, LPSCROLLINFO lpsi)                                                            |
| tii     | int       | FlatSB_GetScrollPos(HWND hwnd, int code)                                                                                 |
| tuiti   | BOOL      | FlatSB_GetScrollProp(HWND hwnd, UINT index, LPINT pValue)                                                                |
| titti   | BOOL      | FlatSB_GetScrollRange(HWND hwnd, int code, LPINT lpMinPos, LPINT lpMaxPos)                                               |
| titii   | int       | FlatSB_SetScrollInfo(HWND hwnd, int fnBar, LPSCROLLINFO lpsi, BOOL fRedraw)                                              |
| tiiii   | int       | FlatSB_SetScrollPos(HWND hwnd, int code, int nPos, BOOL fRedraw)                                                         |
| tuitii  | BOOL      | FlatSB_SetScrollProp(HWND hwnd, UINT index, INT_PTR newValue, BOOL fRedraw)                                              |
| tiiiiii | int       | FlatSB_SetScrollRange(HWND hwnd, int code, int nMinPos, int nMaxPos, BOOL fRedraw)                                       |
| tiii    | BOOL      | FlatSB_ShowScrollBar(HWND hwnd, int code, BOOL fShow)                                                                    |
| ti      | int       | FreeMRUList(_In_ HANDLE hMRU)                                                                                            |
| ttti    | VOID      | GetEffectiveClientRect(HWND hWnd, LPRECT lprc, _In_ const INT *lpInfo)                                                   |
| uh      | LANGID    | GetMUILanguage(void)                                                                                                     |
| tttti   | BOOL      | GetWindowSubclass(_In_ HWND hWnd, _In_ SUBCLASSPROC pfnSubclass, _In_ UINT_PTR uIdSubclass, _Out_ DWORD_PTR *pdwRefData) |
| ttti    | HRESULT   | HIMAGELIST_QueryInterface(_In_ HIMAGELIST himl, _In_ REFIID riid, _Out_ void **ppv)                                      |
|         |           |                                                                                                                          |

|               |            |                                                                                                                                              |
|---------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| ttti          | int        | <code>ImageList_Add(_In_ HIMAGELIST himl, _In_ HBITMAP hbmImage, _In_opt_ HBITMAP hbmMask)</code>                                            |
| ttuii         | int        | <code>ImageList_AddMasked(HIMAGELIST himl, HBITMAP hbmImage, COLORREF crMask)</code>                                                         |
| tiiii         | BOOL       | <code>ImageList_BeginDrag(HIMAGELIST himlTrack, int iTrack, int dxHotspot, int dyHotspot)</code>                                             |
| ttti          | HRESULT    | <code>ImageList_CoCreateInstance(_In_ REFCLSID rclsid, _In_opt_ const IUnknown *punkOuter, _In_ REFIID riid, _Out_ void **ppv)</code>        |
| tituii        | BOOL       | <code>ImageList_Copy(HIMAGELIST himlDst, int iDst, HIMAGELIST himlSrc, int iSrc, UINT uFlags)</code>                                         |
| iiuiiit       | HIMAGELIST | <code>ImageList_Create(int cx, int cy, UINT flags, int cInitial, int cGrow)</code>                                                           |
| ti            | BOOL       | <code>ImageList_Destroy(_In_opt_ HIMAGELIST himl)</code>                                                                                     |
| tiii          | BOOL       | <code>ImageList_DragEnter(HWND hwndLock, int x, int y)</code>                                                                                |
| ti            | BOOL       | <code>ImageList_DragLeave(HWND hwndLock)</code>                                                                                              |
| iii           | BOOL       | <code>ImageList_DragMove(int x, int y)</code>                                                                                                |
| ii            | BOOL       | <code>ImageList_DragShowNoLock(BOOL fShow)</code>                                                                                            |
| tituiii       | BOOL       | <code>ImageList_Draw(HIMAGELIST himl, int i, HDC hdcDst, int x, int y, UINT fStyle)</code>                                                   |
| titiiiiuiuiii | BOOL       | <code>ImageList_DrawEx(HIMAGELIST himl, int i, HDC hdcDst, int x, int y, int dx, int dy, COLORREF rgbBk, COLORREF rgbFg, UINT fStyle)</code> |
| ti            | BOOL       | <code>ImageList_DrawIndirect(IMAGELISTDRAWPARAMS *pimldp)</code>                                                                             |
| tt            | HIMAGELIST | <code>ImageList_Duplicate(HIMAGELIST himl)</code>                                                                                            |
| i             | VOID       | <code>ImageList_EndDrag(void)</code>                                                                                                         |
| tui           | COLORREF   | <code>ImageList_GetBkColor(_In_ HIMAGELIST himl)</code>                                                                                      |
| ttt           | HIMAGELIST | <code>ImageList_GetDragImage(POINT *ppt, POINT *pptHotspot)</code>                                                                           |
| tiuit         | HICON      | <code>ImageList_GetIcon(HIMAGELIST himl, int i, UINT flags)</code>                                                                           |
| ttti          | BOOL       | <code>ImageList_GetIconSize(HIMAGELIST himl, int *cx, int *cy)</code>                                                                        |
| ti            | int        | <code>ImageList_GetImageCount(_In_ HIMAGELIST himl)</code>                                                                                   |
| titi          | BOOL       | <code>ImageList_GetImageInfo(HIMAGELIST himl, int i, IMAGEINFO *pImageInfo)</code>                                                           |
| tsiiuiuiiit   | HIMAGELIST | <code>ImageList_LoadImage(HINSTANCE hi, LPCTSTR lpbmp, int cx, int cGrow, COLORREF crMask, UINT uType, UINT uFlags)</code>                   |
| taiiuiuiiit   | HIMAGELIST | <code>ImageList_LoadImageA(HINSTANCE hi, LPCSTR lpbmp, int cx, int cGrow, COLORREF crMask, UINT uType, UINT uFlags)</code>                   |

|             |            |                                                                                                                                   |
|-------------|------------|-----------------------------------------------------------------------------------------------------------------------------------|
| twiiuiuiuit | HIMAGELIST | <code>ImageList_LoadImageW</code> (HINSTANCE hi, LPCWSTR lpbmp, int cx, int cGrow, COLORREF crMask, UINT uType, UINT uFlags)      |
| titiit      | HIMAGELIST | <code>ImageList_Merge</code> (HIMAGELIST himl1, int i1, HIMAGELIST himl2, int i2, int dx, int dy)                                 |
| tt          | HIMAGELIST | <code>ImageList_Read</code> (LPSTREAM pstm)                                                                                       |
| uittti      | HRESULT    | <code>ImageList_ReadEx</code> (_In_ DWORD dwFlags, _In_ LPSTREAM pstm, _Out_ REFIID riid, _Out_ void **ppv)                       |
| tii         | BOOL       | <code>ImageList_Remove</code> (HIMAGELIST himl, int i)                                                                            |
| titti       | BOOL       | <code>ImageList_Replace</code> (HIMAGELIST himl, int i, HBITMAP hbmImage, HBITMAP hbmMask)                                        |
| titi        | int        | <code>ImageList_ReplaceIcon</code> (_In_ HIMAGELIST himl, _In_ int i, _In_ HICON hicon)                                           |
| tuiui       | COLORREF   | <code>ImageList_SetBkColor</code> (_In_ HIMAGELIST himl, _In_ COLORREF clrBk)                                                     |
| tiii        | BOOL       | <code>ImageList_SetDragCursorImage</code> (HIMAGELIST himlDrag, int iDrag, int dxHotspot, int dyHotspot)                          |
| tiii        | BOOL       | <code>ImageList_SetIconSize</code> (HIMAGELIST himl, int cx, int cy)                                                              |
| tuii        | BOOL       | <code>ImageList_SetImageCount</code> (_In_ HIMAGELIST himl, _In_ UINT uNewCount)                                                  |
| tiii        | BOOL       | <code>ImageList_SetOverlayImage</code> (_In_ HIMAGELIST himl, _In_ int iImage, _In_ int iOverlay)                                 |
| tti         | BOOL       | <code>ImageList_Write</code> (HIMAGELIST himl, LPSTREAM pstm)                                                                     |
| tuiti       | HRESULT    | <code>ImageList_WriteEx</code> (_In_ HIMAGELIST himl, _In_ DWORD dwFlags, _In_ LPSTREAM pstm)                                     |
| i           | VOID       | <code>InitCommonControls</code> (void)                                                                                            |
| ti          | BOOL       | <code>InitCommonControlsEx</code> (_In_ const LPINITCOMMONCONTROLSEX lpInitCtrls)                                                 |
| ti          | BOOL       | <code>InitializeFlatSB</code> (HWND hwnd)                                                                                         |
| uhi         | VOID       | <code>InitMUILanguage</code> (LANGID uiLang)                                                                                      |
| ti6ii       | int        | <code>LBItemFromPt</code> (HWND hLB, POINT pt, BOOL bAutoScroll)                                                                  |
| twiti       | HRESULT    | <code>LoadIconMetric</code> (_In_ HINSTANCE hinst, _In_ PCWSTR pszName, _In_ int lms, _Out_ HICON *phico)                         |
| twiiti      | HRESULT    | <code>LoadIconWithScaleDown</code> (_In_ HINSTANCE hinst, _In_ PCWSTR pszName, _In_ int cx, _In_ int cy, _Out_ HICON *phico)      |
| ti          | BOOL       | <code>MakesDragList</code> (HWND hLB)                                                                                             |
| uittttti    | VOID       | <code>MenuHelp</code> (UINT uMsg, WPARAM wParam, LPARAM lParam, HMENU hMainMenu, HINSTANCE hInst, HWND hwndStatus, LPUINT lpwIDs) |

|           |         |                                                                                                                                                                                                                                               |
|-----------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tt        | INT_PTR | PropertySheet(LPCPROPSHEETHEADER lppsph)                                                                                                                                                                                                      |
| tt        | INT_PTR | PropertySheetA(LPCPROPSHEETHEADER lppsph)                                                                                                                                                                                                     |
| tt        | INT_PTR | PropertySheetW(LPCPROPSHEETHEADER lppsph)                                                                                                                                                                                                     |
| ttti      | BOOL    | RemoveWindowSubclass(_In_ HWND hWnd, _In_ SUBCLASSPROC pfnSubclass, _In_ UINT_PTR uIdSubclass)                                                                                                                                                |
| ttti      | BOOL    | SetWindowSubclass(_In_ HWND hWnd, _In_ SUBCLASSPROC pfnSubclass, _In_ UINT_PTR uIdSubclass, _In_ DWORD_PTR dwRefData)                                                                                                                         |
| ttti      | BOOL    | ShowHideMenuCI(HWND hWnd, UINT_PTR uFlags, LPINT lpInfo)                                                                                                                                                                                      |
| tsi       | BOOL    | Str_SetPtr(_Inout_ LPTSTR *ppszCurrent, LPCWSTR pszNew)                                                                                                                                                                                       |
| twi       | BOOL    | Str_SetPtrW(_Inout_ LPWSTR *ppszCurrent, LPCWSTR pszNew)                                                                                                                                                                                      |
| ttwwwiwti | HRESULT | TaskDialog(_In_ HWND hWndParent, _In_ HINSTANCE hInstance, _In_ PCWSTR pszWindowTitle, _In_ PCWSTR pszMainInstruction, _In_ PCWSTR pszContent, _In_ TASKDIALOG_COMMON_BUTTON_FLAGS dwCommonButtons, _In_ PCWSTR pszIcon, _Out_ int *pnButton) |
| ttti      | HRESULT | TaskDialogIndirect(_In_ const TASKDIALOGCONFIG *pTaskConfig, _Out_opt_ int *pnButton, _Out_opt_ int *pnRadioButton, _Out_opt_ BOOL *pfVerificationFlagChecked)                                                                                |
| ti        | HRESULT | UninitializeFlatSB(HWND hWnd)                                                                                                                                                                                                                 |

## Comdlg32.dll

|        |         |                                                                                 |
|--------|---------|---------------------------------------------------------------------------------|
| ti     | BOOL    | ChooseColor(_Inout_ LPCHOOSECOLOR lpcc)                                         |
| ti     | BOOL    | ChooseColorA(_Inout_ LPCHOOSECOLOR lpcc)                                        |
| ti     | BOOL    | ChooseColorW(_Inout_ LPCHOOSECOLOR lpcc)                                        |
| ti     | BOOL    | ChooseFont(_Inout_ LPCHOOSEFONT lpfc)                                           |
| ti     | BOOL    | ChooseFontA(_Inout_ LPCHOOSEFONT lpfc)                                          |
| ti     | BOOL    | ChooseFontW(_Inout_ LPCHOOSEFONT lpfc)                                          |
| ui     | DWORD   | CommDlgExtendedError(void)                                                      |
| tt     | HWND    | FindText(_In_ LPFINDREPLACE lpfr)                                               |
| tt     | HWND    | FindTextA(_In_ LPFINDREPLACE lpfr)                                              |
| tt     | HWND    | FindTextW(_In_ LPFINDREPLACE lpfr)                                              |
| ssuhh  | SHORT   | GetFileTitle(_In_ LPCTSTR lpszFile, _Out_ LPTSTR lpszTitle, _In_ WORD cchSize)  |
| aauihh | SHORT   | GetFileTitleA(_In_ LPCSTR lpszFile, _Out_ LPSTR lpszTitle, _In_ WORD cchSize)   |
| wwuihh | SHORT   | GetFileTitleW(_In_ LPCWSTR lpszFile, _Out_ LPWSTR lpszTitle, _In_ WORD cchSize) |
| ti     | BOOL    | GetOpenFileName(_Inout_ LPOPENFILENAME lpofn)                                   |
| ti     | BOOL    | GetOpenFileNameA(_Inout_ LPOPENFILENAME lpofn)                                  |
| ti     | BOOL    | GetOpenFileNameW(_Inout_ LPOPENFILENAME lpofn)                                  |
| ti     | BOOL    | GetSaveFileName(_Inout_ LPOPENFILENAME lpofn)                                   |
| ti     | BOOL    | GetSaveFileNameA(_Inout_ LPOPENFILENAME lpofn)                                  |
| ti     | BOOL    | GetSaveFileNameW(_Inout_ LPOPENFILENAME lpofn)                                  |
| ti     | BOOL    | PageSetupDlg(_Inout_ LPPAGESETUPDLG lppsd)                                      |
| ti     | BOOL    | PageSetupDlgA(_Inout_ LPPAGESETUPDLG lppsd)                                     |
| ti     | BOOL    | PageSetupDlgW(_Inout_ LPPAGESETUPDLG lppsd)                                     |
| ti     | BOOL    | PrintDlg(_Inout_ LPPRINTDLG lppd)                                               |
| ti     | BOOL    | PrintDlgA(_Inout_ LPPRINTDLG lppd)                                              |
| ti     | HRESULT | PrintDlgEx(_Inout_ LPPRINTDLGEX lppd)                                           |
| ti     | HRESULT | PrintDlgExA(_Inout_ LPPRINTDLGEX lppd)                                          |
| ti     | HRESULT | PrintDlgExW(_Inout_ LPPRINTDLGEX lppd)                                          |
| ti     | BOOL    | PrintDlgW(_Inout_ LPPRINTDLG lppd)                                              |
| tt     | HWND    | ReplaceText(_Inout_ LPFINDREPLACE lpfr)                                         |

|    |      |                                                           |
|----|------|-----------------------------------------------------------|
| tt | HWND | <a href="#">ReplaceTextA</a> (_Inout_ LPFINDREPLACE lpfr) |
| tt | HWND | <a href="#">ReplaceTextW</a> (_Inout_ LPFINDREPLACE lpfr) |

# Crypt32.dll

|            |      |                                                                                                                                                                                                  |
|------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiti     | BOOL | <a href="#">CertAddCertificateContextToStore</a> (hCertStore, _In_ PCCERT_CONTEXT *ppStoreContext, DWORD dwAddDisposition, _Out_ PCCERT_CONTEXT *ppStoreContext)                                 |
| ttuiti     | BOOL | <a href="#">CertAddCertificateLinkToStore</a> (hCertStore, _In_ PCCERT_CONTEXT *ppStoreContext, DWORD dwAddDisposition, _Out_ PCCERT_CONTEXT *ppStoreContext)                                    |
| ttuiti     | BOOL | <a href="#">CertAddCRLContextToStore</a> (_In_ hCertStore, _In_ PCCRL_CONTEXT *ppStoreContext, DWORD dwAddDisposition, _Out_ PCCRL_CONTEXT *ppStoreContext)                                      |
| ttuiti     | BOOL | <a href="#">CertAddCRLLinkToStore</a> (_In_ hCertStore, _In_ PCCRL_CONTEXT *ppStoreContext, _In_ PCCRL_CONTEXT pCrlContext, dwAddDisposition, _Out_opt_ PCCRL_CONTEXT *ppStoreContext)           |
| ttuiti     | BOOL | <a href="#">CertAddCTLContextToStore</a> (_In_ hCertStore, _In_ PCCTL_CONTEXT *ppStoreContext, DWORD dwAddDisposition, _Out_ PCCTL_CONTEXT *ppStoreContext)                                      |
| ttuiti     | BOOL | <a href="#">CertAddCTLLinkToStore</a> (_In_ hCertStore, _In_ PCCTL_CONTEXT *ppStoreContext, _In_ PCCTL_CONTEXT pCtlContext, dwAddDisposition, _Out_opt_ PCCTL_CONTEXT *ppStoreContext)           |
| tuituiuiti | BOOL | <a href="#">CertAddEncodedCertificateToStore</a> (hCertStore, _In_ DWORD dwCertEncodingType, const BYTE *pbCertEncoded, _In_ DWORD dwCertEncodingType, _Out_opt_ PCCERT_CONTEXT *ppStoreContext) |
| atuii      | BOOL | <a href="#">CertAddEncodedCertificateToSystemStore</a> (szCertStoreName, _In_ const BYTE *pbCertEncoded, _In_ DWORD dwCertEncodingType)                                                          |
| atuii      | BOOL | <a href="#">CertAddEncodedCertificateToSystemStore</a> (szCertStoreName, _In_ const BYTE *pbCertEncoded, _In_ DWORD dwCertEncodingType)                                                          |
| atuii      | BOOL | <a href="#">CertAddEncodedCertificateToSystemStore</a> (LPCSTR szCertStoreName, _In_ const BYTE *pbCertEncoded, _In_ DWORD dwCertEncodingType)                                                   |
| tuituiuiti | BOOL | <a href="#">CertAddEncodedCRLToStore</a> (_In_ hCertStore, _In_ DWORD dwCrlEncodingType, const BYTE *pbCrlEncoded, _In_ DWORD dwCrlEncodingType, _Out_opt_ PCCRL_CONTEXT *ppStoreContext)        |

|              |        |                                                                                                                                                                                        |
|--------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |        | PCCRL_CONTEXT *ppCrlCon                                                                                                                                                                |
| tuituiuti    | BOOL   | CertAddEncodedCTLToStore(_In_ HCERTSTORE hCertStore, _In_ DWORD dwM...<br>_In_ const BYTE *pbCtlEncode...<br>cbCtlEncoded, _In_ DWORD dw...<br>_Out_opt_ PCCTL_CONTEXT                 |
| tai          | BOOL   | CertAddEnhancedKeyUsageIde...<br>PCCERT_CONTEXT pCertCon...<br>pszUsageIdentifier)                                                                                                     |
| ti           | VOID   | CertAddRefServerOcsResponse...<br>HCERT_SERVER_OCSP_RES...<br>hServerOcsResponse)                                                                                                      |
| ti           | VOID   | CertAddRefServerOcsResponse...<br>PCCERT_SERVER_OCSP_RES...<br>pServerOcsResponseContext)                                                                                              |
| ttuiuiuiutti | BOOL   | CertAddSerializedElementToSto...<br>hCertStore, _In_ const BYTE *p...<br>cbElement, _In_ DWORD dwAc...<br>DWORD dwFlags, _In_ DWOR...<br>_Out_ DWORD *pdwContextTy...<br>**ppvContext) |
| ttuiuii      | BOOL   | CertAddStoreToCollection(_In_...<br>hCollectionStore, _In_opt_ HCE...<br>_In_ DWORD dwUpdateFlag, _                                                                                    |
| uia          | LPCSTR | CertAlgIdToOID(_In_ DWORD                                                                                                                                                              |
| tuii         | VOID   | CertCloseServerOcsResponse(_...<br>HCERT_SERVER_OCSP_RES...<br>hServerOcsResponse, _In_ DW                                                                                             |
| tuii         | BOOL   | CertCloseStore(_In_ HCERTST...<br>DWORD dwFlags)                                                                                                                                       |
| uitti        | BOOL   | CertCompareCertificate(_In_ DV...<br>dwCertEncodingType, _In_ PCE...<br>PCERT_INFO pCertId2)                                                                                           |
| uitti        | BOOL   | CertCompareCertificateName(_I...<br>dwCertEncodingType, _In_ PCE...<br>pCertName1, _In_ PCERT_NAM                                                                                      |
| titi         | BOOL   | CertCompareIntegerBlob(_In_ P...<br>pInt1, _In_ PCRYPT_INTEGER                                                                                                                         |
| uitti        | BOOL   | CertComparePublicKeyInfo(_In...<br>dwCertEncodingType, _In_ PCE...<br>pPublicKey1, _In_ PCERT_PUE...<br>pPublicKey2)                                                                   |
| tuiuti       | BOOL   | CertControlStore(_In_ HCERTS...<br>DWORD dwFlags, _In_ DWOR...<br>void *pvCtrlPara)                                                                                                    |

|            |                      |                                                                                                                                                                                                                                                                                                                                 |
|------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tti        | BOOL                 | <a href="#">CertCreateCertificateChainEngine</a> (<br>PCERT_CHAIN_ENGINE_CONFIG<br>HCERTCHAINENGINE *phCh                                                                                                                                                                                                                       |
| uituit     | PCCERT_CONTEXT       | <a href="#">CertCreateCertificateContext</a> (<br>_In_ dwCertEncodingType, _In_ const<br>_In_ DWORD cbCertEncoded)                                                                                                                                                                                                              |
| uiuituiuit | void*                | <a href="#">CertCreateContext</a> (<br>_In_ DWORD<br>_In_ DWORD dwEncodingType, _In_<br>_In_ DWORD cbEncoded, _In_<br>_In_opt_ PCERT_CREATE_CONTEXT<br>pCreatePara)                                                                                                                                                             |
| uituit     | PCCRL_CONTEXT        | <a href="#">CertCreateCRLContext</a> (<br>_In_ DV<br>_In_ dwCertEncodingType, _In_ const<br>_In_ DWORD cbCrlEncoded)                                                                                                                                                                                                            |
| uituit     | PCCTL_CONTEXT        | <a href="#">CertCreateCTLContext</a> (<br>_In_ DV<br>_In_ dwMsgAndCertEncodingType, _<br>_In_ *pbCtlEncoded, _In_ DWORD c                                                                                                                                                                                                       |
| tuituitti  | BOOL                 | <a href="#">CertCreateCTLEntryFromCertif</a><br>_In_ PCCERT_CONTEXT pCertCon<br>_In_ cOptAttr, _In_ PCRYPT_ATTRI<br>_In_ DWORD dwFlags, _In_ void *p<br>_In_ PCTL_ENTRY pCtlEntry, _Inou                                                                                                                                        |
| ttuittttt  | PCCERT_CONTEXT       | <a href="#">CertCreateSelfSignCertificate</a> (<br>_In_ HCRYPTPROV_OR_NCRYPT_<br>_In_ hCryptProvOrNCryptKey, _In_<br>_In_ pSubjectIssuerBlob, _In_ DWO<br>_In_ PCRYPT_KEY_PROV_INFO p<br>_In_ PCRYPT_ALGORITHM_IDEN<br>_In_ pSignatureAlgorithm, _In_opt_<br>_In_ pStartTime, _In_opt_ PSYSTEM<br>_In_ PCERT_EXTENSIONS pExtens |
| ti         | BOOL                 | <a href="#">CertDeleteCertificateFromStore</a> (<br>_In_ pCertContext)                                                                                                                                                                                                                                                          |
| ti         | BOOL                 | <a href="#">CertDeleteCRLFromStore</a> (<br>_In_<br>_In_ pCrlContext)                                                                                                                                                                                                                                                           |
| ti         | BOOL                 | <a href="#">CertDeleteCTLFromStore</a> (<br>_In_<br>_In_ pCtlContext)                                                                                                                                                                                                                                                           |
| tt         | PCCERT_CHAIN_CONTEXT | <a href="#">CertDuplicateCertificateChain</a> (<br>_In_<br>_In_ PCCERT_CHAIN_CONTEXT p                                                                                                                                                                                                                                          |
| tt         | PCCERT_CONTEXT       | <a href="#">CertDuplicateCertificateContext</a> (<br>_In_<br>_In_ pCertContext)                                                                                                                                                                                                                                                 |
| tt         | PCCRL_CONTEXT        | <a href="#">CertDuplicateCRLContext</a> (<br>_In_<br>_In_ pCrlContext)                                                                                                                                                                                                                                                          |
| tt         | PCCTL_CONTEXT        | <a href="#">CertDuplicateCTLContext</a> (<br>_In_<br>_In_ pCtlContext)                                                                                                                                                                                                                                                          |
|            |                      |                                                                                                                                                                                                                                                                                                                                 |

|            |                      |                                                                                                                                                                                                                            |
|------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tt         | HCERTSTORE           | CertDuplicateStore(_In_ HCERTSTORE)                                                                                                                                                                                        |
| tuiui      | DWORD                | CertEnumCertificateContextProperties(PCCERT_CONTEXT pCertContext, DWORD dwPropId)                                                                                                                                          |
| ttt        | PCCERT_CONTEXT       | CertEnumCertificatesInStore(_In_ HCERTSTORE hCertStore, _In_ PCCERT_CONTEXT pPrevCertContext)                                                                                                                              |
| tuiui      | DWORD                | CertEnumCRLContextProperties(pCrlContext, _In_ DWORD dwPropId)                                                                                                                                                             |
| ttt        | PCCRL_CONTEXT        | CertEnumCRLsInStore(_In_ HCERTSTORE hCertStore, _In_ PCCRL_CONTEXT pPrevCrlContext)                                                                                                                                        |
| tuiui      | DWORD                | CertEnumCTLContextProperties(pCtlContext, _In_ DWORD dwPropId)                                                                                                                                                             |
| ttt        | PCCTL_CONTEXT        | CertEnumCTLsInStore(_In_ HCERTSTORE hCertStore, _In_ PCCTL_CONTEXT pPrevCtlContext)                                                                                                                                        |
| tuitti     | BOOL                 | CertEnumPhysicalStore(_In_ void *pvSystemStoreLocationParameter, _In_ DWORD dwFlags, _In_ void *pvFindPara, PFN_CERT_ENUM_PHYSICAL_STORE pfnEnum)                                                                          |
| tttti      | BOOL                 | CertEnumSubjectInSortedCTL(_In_ PCCTL_CONTEXT pCtlContext, _Inout_ void **ppvSystemStoreLocationParameter, PCRYPT_DER_BLOB pSubjectBlob, PCRYPT_DER_BLOB pEncodedSubjectBlob, PFN_CERT_ENUM_SUBJECT_IN_SORTED_CTL pfnEnum) |
| uittti     | BOOL                 | CertEnumSystemStore(_In_ DWORD dwFlags, _In_ void *pvSystemStoreLocationParameter, PFN_CERT_ENUM_SYSTEM_STORE pfnEnum)                                                                                                     |
| uitti      | BOOL                 | CertEnumSystemStoreLocation(_In_ void *pvArg, _In_ PFN_CERT_ENUM_SYSTEM_STORE pfnEnum)                                                                                                                                     |
| auiuit     | PCRYPT_ATTRIBUTE     | CertFindAttribute(_In_ LPCSTR pszAttr, _In_ CRYPT_ATTRIBUTE_TYPE dwType)                                                                                                                                                   |
| ttuitti    | BOOL                 | CertFindCertificateInCRL(_In_ PCRL_CONTEXT pCrlContext, _In_ PCCRL_CONTEXT pCertContext, _In_ DWORD dwFlags, _In_opt_ void *pvFindPara, PCRL_ENTRY *ppCrlEntry)                                                            |
| tuiuiuittt | PCCERT_CONTEXT       | CertFindCertificateInStore(_In_ HCERTSTORE hCertStore, _In_ DWORD dwCertEncodingType, _In_ DWORD dwFindFlags, _In_ DWORD dwCertUsage, const void *pvFindPara, _In_ PCCERT_CONTEXT pPrevCertContext)                        |
| tuiuiuittt | PCCERT_CHAIN_CONTEXT | CertFindChainInStore(_In_ HCERTSTORE hCertStore, _In_ DWORD dwCertEncodingType, _In_ DWORD dwFindFlags, _In_ DWORD dwCertUsage, const void *pvFindPara, _In_ PCCERT_CHAIN_CONTEXT pPrevChainContext)                       |
|            |                      | CertFindCRLInStore(_In_ HCERTSTORE hCertStore, _In_ PCCRL_CONTEXT pPrevCrlContext)                                                                                                                                         |

|            |                 |                                                                                                                                                                                                                  |
|------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiuittt | PCCRL_CONTEXT   | DWORD dwEncodingType, _In_ _In_ DWORD dwFindType, _In_ _In_ PCCRL_CONTEXT pPrev                                                                                                                                  |
| tuiuiuittt | PCCTL_CONTEXT   | CertFindCTLInStore(_In_ HCEI<br>DWORD dwMsgAndCertEncod<br>dwFindFlags, _In_ DWORD dw<br>*pvFindPara, _In_ PCCTL_CON                                                                                             |
| auiuit     | PCERT_EXTENSION | CertFindExtension(_In_ LPCST<br>cExtensions, _In_ CERT_EXTE                                                                                                                                                      |
| att        | PCERT_RDN_ATTR  | CertFindRDNAtr(_In_ LPCSTP<br>PCERT_NAME_INFO pName)                                                                                                                                                             |
| uiuittuit  | PCTL_ENTRY      | CertFindSubjectInCTL(_In_ DW<br>_In_ DWORD dwSubjectType, _<br>PCCTL_CONTEXT pCtlContext                                                                                                                         |
| ttuitti    | BOOL            | CertFindSubjectInSortedCTL(_I<br>pSubjectIdentifier, _In_ PCCTL<br>_In_ DWORD dwFlags, _In_ vo<br>PCRYPT_DER_BLOB pEncode                                                                                        |
| ti         | VOID            | CertFreeCertificateChain(_In_<br>PCCERT_CHAIN_CONTEXT p                                                                                                                                                          |
| ti         | VOID            | CertFreeCertificateChainEngine<br>HCERTCHAINENGINE hChain                                                                                                                                                        |
| ti         | VOID            | CertFreeCertificateChainList(_In_<br>PCCERT_CHAIN_CONTEXT *                                                                                                                                                      |
| ti         | BOOL            | CertFreeCertificateContext(_In_<br>pCertContext)                                                                                                                                                                 |
| ti         | BOOL            | CertFreeCRLContext(_In_ PCC<br>pCrlContext)                                                                                                                                                                      |
| ti         | BOOL            | CertFreeCTLContext(_In_ PCC<br>pCtlContext)                                                                                                                                                                      |
| ti         | int             | CertFreeServerOcspResponseCo<br>PCCERT_SERVER_OCSP_RES<br>pServerOcspResponseContext)                                                                                                                            |
| ttttuitti  | BOOL            | CertGetCertificateChain(_In_ op<br>hChainEngine, _In_ PCCERT_C<br>_In_opt_ LPFILETIME pTime, _<br>hAdditionalStore, _In_ PCERT_<br>pChainPara, _In_ DWORD dwF<br>pvReserved, _Out_ PCCERT_CI<br>*ppChainContext) |
| tuitti     | BOOL            | CertGetCertificateContextProp<br>PCCERT_CONTEXT pCertCon<br>dwPropId, _Out_ void *pvData,<br>*pcbData)                                                                                                           |
|            |                 | CertGetCRLContextProperty(_I                                                                                                                                                                                     |

|            |                                     |                                                                                                                                                                                                         |
|------------|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti     | BOOL                                | <a href="#">pCrlContext, _In_ DWORD dwFlags, _Inout_ LPVOID pvData, _Inout_ DWORD *pcbData</a>                                                                                                          |
| tttt       | PCCRL_CONTEXT                       | <a href="#">CertGetCRLFromStore(_In_ HCERT_CONTEXT hCertStore, _In_opt_ PCCERT_CONTEXT pCrlContext, PCCRL_CONTEXT pPrevCrlContext, _Inout_ DWORD *pdwFlags)</a>                                         |
| tuitti     | BOOL                                | <a href="#">CertGetCTLContextProperties(_In_ pCtlContext, _In_ DWORD dwFlags, _Inout_ LPVOID pvData, _Inout_ DWORD *pcbData)</a>                                                                        |
| tuitti     | BOOL                                | <a href="#">CertGetEnhancedKeyUsage(_In_ pCertContext, _In_ DWORD dwFlags, _Inout_ PCERT_ENHKEY_USAGE pUsage, _Inout_ LPVOID *pcbUsage)</a>                                                             |
| uittui     | BOOL                                | <a href="#">CertGetIntendedKeyUsage(_In_ pCertContext, _In_ DWORD dwCertEncodingType, _In_ PCERT_ENHKEY_USAGE *pbKeyUsage, _Inout_ LPVOID pvData, _Inout_ DWORD *pcbData)</a>                           |
| tttt       | PCCERT_CONTEXT                      | <a href="#">CertGetIssuerCertificateFromStore(_In_ hCertStore, _In_ PCCERT_CONTEXT pCertificate, _In_opt_ PCCERT_CONTEXT pCrlContext, _Inout_ DWORD *pdwFlags)</a>                                      |
| tuiuitsui  | DWORD                               | <a href="#">CertGetNameString(_In_ PCERT_CONTEXT pCertContext, _In_ DWORD dwCertEncodingType, _In_ DWORD dwFlags, _In_ void *pvTypeParameters, _In_ LPCTSTR pszNameString, _In_ DWORD dwNameSpace)</a>  |
| tuiuitaiui | DWORD                               | <a href="#">CertGetNameStringA(_In_ PCERT_CONTEXT pCertContext, _In_ DWORD dwCertEncodingType, _In_ DWORD dwFlags, _In_ void *pvTypeParameters, _In_ LPCTSTR pszNameString, _In_ DWORD dwNameSpace)</a> |
| tuiuitwui  | DWORD                               | <a href="#">CertGetNameStringW(_In_ PCERT_CONTEXT pCertContext, _In_ DWORD dwCertEncodingType, _In_ DWORD dwFlags, _In_ void *pvTypeParameters, _In_ LPCTSTR pszNameString, _In_ DWORD dwNameSpace)</a> |
| uitui      | DWORD                               | <a href="#">CertGetPublicKeyLength(_In_ pCertContext, _In_ DWORD dwCertEncodingType, _In_ PCERT_PUBLIC_KEY_INFO pPublicKey)</a>                                                                         |
| tuitt      | PCCERT_SERVER_OCSP_RESPONSE_CONTEXT | <a href="#">CertGetServerOcspResponseContext(_In_ HCERT_SERVER_OCSP_RESPONSE_CONTEXT hServerOcspResponse, _In_ DWORD dwFlags, _Inout_ LPVOID pvReserved)</a>                                            |
| tuitti     | BOOL                                | <a href="#">CertGetStoreProperty(_In_ HCERT_CONTEXT hCertStore, _In_ DWORD dwPropId, _Out_ LPVOID pvData, _Inout_ DWORD *pcbData)</a>                                                                   |
| tuitt      | PCCERT_CONTEXT                      | <a href="#">CertGetSubjectCertificateFromStore(_In_ hCertStore, _In_ DWORD dwCertEncodingType, _In_ PCERT_INFO pCertId)</a>                                                                             |
|            |                                     | <a href="#">CertGetValidUsages(_In_ DWORD dwCertEncodingType, _In_ PCERT_ENHKEY_USAGE *pbKeyUsage, _Inout_ LPVOID pvData, _Inout_ DWORD *pcbData)</a>                                                   |

|            |                            |                                                                                                                              |
|------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------|
| uitttti    | BOOL                       | PCCERT_CONTEXT *rghCerts<br>_Out_ LPSTR *rghOIDs, _Inout                                                                     |
| uiuitti    | BOOL                       | CertIsRDNAutrsInCertificateName<br>dwCertEncodingType, _In_ DWORD<br>PCERT_NAME_BLOB pCertName,<br>pRDN)                     |
| ttuiti     | BOOL                       | CertIsValidCRLForCertificate(_<br>pCert, _In_ PCCRL_CONTEXT<br>dwFlags, _In_ void *pvReserved)                               |
| uituisuiui | DWORD                      | CertNameToStr(_In_ DWORD dw<br>PCERT_NAME_BLOB pName,<br>_Out_ LPTSTR psz, _In_ DWORD)                                       |
| uituiauiui | DWORD                      | CertNameToStrA(_In_ DWORD<br>_In_ PCERT_NAME_BLOB pName,<br>dwStrType, _Out_ LPSTR psz, _                                    |
| uituiwuiui | DWORD                      | CertNameToStrW(_In_ DWORD<br>_In_ PCERT_NAME_BLOB pName,<br>dwStrType, _Out_ LPWSTR psz)                                     |
| aiui       | DWORD                      | CertOIDToAlgId(_In_ LPCSTR                                                                                                   |
| tuitt      | HCERT_SERVER_OCSP_RESPONSE | CertOpenServerOcapResponse(_<br>PCCERT_CHAIN_CONTEXT pChain,<br>DWORD dwFlags, _Reserved_)                                   |
| auituitt   | HCERTSTORE                 | CertOpenStore(_In_ LPCSTR lp<br>DWORD dwMsgAndCertEncod<br>HCRYPTPROV_LEGACY hCryptProv,<br>dwFlags, _In_ const void *pvPara |
| tst        | HCERTSTORE                 | CertOpenSystemStore(_In_ HC<br>hprov, _In_ LPTCSTR szSubsystem                                                               |
| tat        | HCERTSTORE                 | CertOpenSystemStoreA(_In_ HC<br>hprov, _In_ LPCSTR szSubsystem                                                               |
| twt        | HCERTSTORE                 | CertOpenSystemStoreW(_In_ HC<br>hprov, _In_ LPCWSTR szSubsystem                                                              |
| uitsuiui   | DWORD                      | CertRDNValueToStr(_In_ DW<br>PCERT_RDN_VALUE_BLOB pRDNValue,<br>psz, _In_ DWORD csz)                                         |
| uitaiuiui  | DWORD                      | CertRDNValueToStrA(_In_ DW<br>PCERT_RDN_VALUE_BLOB pRDNValue,<br>_In_ DWORD csz)                                             |
| uitwuiui   | DWORD                      | CertRDNValueToStrW(_In_ DW<br>PCERT_RDN_VALUE_BLOB pRDNValue,<br>psz, _In_ DWORD csz)                                        |
| tuiwtti    | BOOL                       | CertRegisterPhysicalStore(_In_<br>_In_ DWORD dwFlags, _In_ LP<br>_In_ PCERT_PHYSICAL_STORE<br>void *pvReserved)              |

|             |      |                                                                                                                                                                                                                               |
|-------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti      | BOOL | <a href="#">CertRegisterSystemStart</a> (_In_ c<br>_In_ DWORD dwFlags, _In_<br>PCCERT_SYSTEM_STORE_INF<br>*pvReserved)                                                                                                        |
| tai         | BOOL | <a href="#">CertRemoveEnhancedKeyUsage</a><br>PCCERT_CONTEXT pCertCon<br>pszUsageIdentifier)                                                                                                                                  |
| tti         | VOID | <a href="#">CertRemoveStoreFromCollectio</a><br>hCollectionStore, _In_ HCERTS                                                                                                                                                 |
| tauuiuittti | BOOL | <a href="#">CertRetrieveLogoOrBiometricIn</a><br>PCCERT_CONTEXT pCertCon<br>lpzLogoOrBiometricType, _In_<br>dwRetrievalFlags, _In_ DWOR<br>DWORD dwFlags, _Reserved_<br>BYTE **ppbData, _Out_ DWO<br>LPWSTR *ppwszMimeType)   |
| tuiuiituii  | BOOL | <a href="#">CertSaveStore</a> (_In_ HCERTSTO<br>DWORD dwMsgAndCertEncod<br>dwSaveAs, _In_ DWORD dwSa<br>*pvSaveToPara, _In_ DWORD c                                                                                           |
| tuituittti  | BOOL | <a href="#">CertSelectCertificateChain</a> (_In_<br>pSelectionContext, _In_ DWOR<br>PCCERT_SELECT_CHAIN_PA<br>_In_ DWORD cCriteria, _In_op<br>PCCERT_SELECT_CRITERIA<br>HCERTSTORE hStore, _Out_ P<br>_Out_ PCCERT_CHAIN_CONT |
| tuitti      | BOOL | <a href="#">CertSerializeCertificateStoreEle</a><br>PCCERT_CONTEXT pCertCon<br>dwFlags, _Out_ BYTE *pbElem<br>*pcbElement)                                                                                                    |
| tuitti      | BOOL | <a href="#">CertSerializeCRLStoreElement</a> (<br>pCrlContext, _In_ DWORD dwI<br>*pbElement, _Inout_ DWORD *                                                                                                                  |
| tuitti      | BOOL | <a href="#">CertSerializeCTLStoreElement</a> (<br>pCtlContext, _In_ DWORD dwE<br>*pbElement, _Inout_ DWORD *                                                                                                                  |
| ttuii       | BOOL | <a href="#">CertSetCertificateContextProper</a><br>PCCERT_CONTEXT pCertCon<br>pCtlEntry, _In_ DWORD dwFla                                                                                                                     |
| tuiuiti     | BOOL | <a href="#">CertSetCertificateContextProper</a><br>PCCERT_CONTEXT pCertCon<br>dwPropId, _In_ DWORD dwFla<br>*pvData)                                                                                                          |
| tuiuiti     | BOOL | <a href="#">CertSetCRLContextProperty</a> (_In_<br>pCrlContext, _In_ DWORD dwI                                                                                                                                                |

|            |      |                                                                                                                                                                                                                                          |
|------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |      | dwFlags, _In_ const void *pvData)                                                                                                                                                                                                        |
| tuiuiti    | BOOL | CertSetCTLContextProperty(_In_ pCtlContext, _In_ DWORD dwFlags, _In_ const void *pvData)                                                                                                                                                 |
| tti        | BOOL | CertSetEnhancedKeyUsage(_In_ pCertContext, _In_ PCERT_ENHANCED_KEY_USAGE pKeyUsage)                                                                                                                                                      |
| tuiuiti    | BOOL | CertSetStoreProperty(_In_ HCE_STORE_PROPERTY_ID dwPropId, _In_ DWORD dwValue, _In_ const void *pvData)                                                                                                                                   |
| uisuitttsi | BOOL | CertSirToName(_In_ DWORD dwCertEncodingType, _In_ LPCTSTR pszX500, _In_ DWORD dwReserved, _Out_ BYTE *pvReserved, _Out_ DWORD *pcbEncoded, _Out_ DWORD *pcbDecoded)                                                                      |
| uiauitttai | BOOL | CertSirToNameA(_In_ DWORD dwCertEncodingType, _In_ LPCSTR pszX500, _In_ DWORD dwReserved, _Out_ void *pvReserved, _Out_ DWORD *pcbEncoded, _Out_ DWORD *pcbDecoded, _Out_ ppszError)                                                     |
| uiwuitttwi | BOOL | CertSirToNameW(_In_ DWORD dwCertEncodingType, _In_ LPCWSTR pszX500, _In_ DWORD dwReserved, _Out_ void *pvReserved, _Out_ DWORD *pcbEncoded, _Out_ DWORD *pcbDecoded, _Out_ ppszError)                                                    |
| tuiwi      | BOOL | CertUnregisterPhysicalStore(_In_ void *pvSystemStore, _In_ DWORD dwFlags, _In_ PWSTR pwszStoreName)                                                                                                                                      |
| tuii       | BOOL | CertUnregisterSystemStore(_In_ void *pvSystemStore, _In_ DWORD dwFlags)                                                                                                                                                                  |
| attti      | BOOL | CertVerifyCertificateChainPolicy(pszPolicyOID, _In_ PCCERT_CHAIN_CONTEXT pChainContext, _In_ PCERT_CHAIN_POLICY_PARA pPolicyPara, _Inout_ PCERT_CHAIN_POLICY_STATUS pPolicyStatus)                                                       |
| uituiti    | BOOL | CertVerifyCRLRevocation(_In_ DWORD dwCertEncodingType, _In_ PCERT_CRL_CONTEXT pCrlContext, _In_ PCRL_INFO pCrlInfo, _In_ PCRL_REVOCATION_PARA pRevocationPara, _Inout_ PCERT_CRL_REVOCATION_STATUS pRevocationStatus)                    |
| ttui       | LONG | CertVerifyCRLTimeValidity(_In_ PCERT_CRL_CONTEXT pTimeToVerify, _In_ PCRL_INFO pCrlInfo)                                                                                                                                                 |
| uiuittuiti | BOOL | CertVerifyCTLUsage(_In_ DWORD dwCertEncodingType, _In_ DWORD dwSubjectType, _In_ PCTL_USAGE pSubjectUsage, _In_opt_ PCTL_VERIFY_USAGE_PARA pVerifyUsagePara, _Inout_ PCTL_VERIFY_USAGE_STATUS pVerifyUsageStatus)                        |
|            |      | CertVerifyRevocation(_In_ DWORD dwCertEncodingType, _In_ DWORD dwRevType, _In_ PCERT_CRL_CONTEXT pCrlContext, _In_ PCRL_INFO pCrlInfo, _In_ PCRL_REVOCATION_PARA pRevocationPara, _Inout_ PCERT_CRL_REVOCATION_STATUS pRevocationStatus) |

|                    |      |                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiuituitti       | BOOL | PVOID rgpvContext[], _In_ DWORD PCERT_REVOCATION_PARAMETERS, PCERT_REVOCATION_STATUS                                                                                                                                                                                                                                                                                                                  |
| ttti               | BOOL | CertVerifySubjectCertificateContext(PCCERT_CONTEXT pSubject, PCCERT_CONTEXT pIssuer, _Out_ *pdwFlags)                                                                                                                                                                                                                                                                                                 |
| ttui               | LONG | CertVerifyTimeValidity(_In_ LPCTSTR pTime, _In_ PCERT_INFO pCertInfo)                                                                                                                                                                                                                                                                                                                                 |
| tti                | BOOL | CertVerifyValidityNesting(_In_ PCERT_INFO pSubjectInfo, _In_ PCERT_INFO pIssuerInfo)                                                                                                                                                                                                                                                                                                                  |
| tuitttti           | BOOL | CryptAcquireCertificatePrivateKey(PCCERT_CONTEXT pCert, _In_ DWORD dwFlags, _In_opt_ void *pvParameters, _Out_ HCRYPTPROV_OR_NCRYPT_KEY_HANDLE *phCryptProvOrNCryptKey, _Out_ *pdwKeySpec, _Out_ BOOL *pfCallerFreeProvOrNCryptKey)                                                                                                                                                                   |
| tuiuisti           | BOOL | CryptBinaryToString(_In_ const BYTE *pbBinary, _In_ DWORD cbBinary, _In_ DWORD dwEncoding, LPWSTR pszString, _Inout_ DWORD *pdwStringLength)                                                                                                                                                                                                                                                          |
| tuiuiati           | BOOL | CryptBinaryToStringA(_In_ const BYTE *pbBinary, _In_ DWORD cbBinary, _In_ DWORD dwEncoding, LPSTR pszString, _Inout_ DWORD *pdwStringLength)                                                                                                                                                                                                                                                          |
| tuiuiwti           | BOOL | CryptBinaryToStringW(_In_ const BYTE *pbBinary, _In_ DWORD cbBinary, _In_ DWORD dwEncoding, LPWSTR pszString, _Inout_ DWORD *pdwStringLength)                                                                                                                                                                                                                                                         |
| uiatuiuittti       | BOOL | CryptCreateKeyIdentifierFromCertificateObject(dwCertEncodingType, _In_ LPCWSTR pszSubjectName, const PUBLICKEYSTRUC *pPubKeyStruc, _In_ DWORD dwFlags, _Out_ BYTE *pbKeyIdentifier, _Out_ *pcbKeyIdentifierLength, _Out_ *pcbHash)                                                                                                                                                                    |
| uittuituiuitttttti | BOOL | CryptDecodeMessage(_In_ DWORD dwMsgType, _In_ PCRYPT_DECRYPT_PARAMETERS pDecryptPara, _In_ PCRYPT_VERIFY_PARAMETERS pVerifyPara, _In_ DWORD dwSize, _In_ const BYTE *pbEncodedBlob, _In_ DWORD dwPrevInnerContentType, _In_ DWORD *pdwMsgType, _Out_ *pdwInnerContentType, _Out_opt_ *pdwInnerContentLength, _Inout_opt_ DWORD *pcbDecoded, PCCERT_CONTEXT *ppXchgCert, PCCERT_CONTEXT *ppSignerCert) |
| uiatuiuitti        | BOOL | CryptDecodeObject(_In_ DWORD dwMsgType, _In_ LPCSTR lpszStructType, _In_ const BYTE *pbEncoded, _In_ DWORD cbEncoded, _In_ DWORD dwFlags, _Out_ *pdwDecodedLength, _Out_ *ppvDecoded)                                                                                                                                                                                                                 |

|              |      |                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |      | dwFlags, _Out_ void *pvStructInfo, _Out_ PCCERT_CONTEXT *pcbStructInfo)                                                                                                                                                                                                                                                                                                  |
| uiatuiuittti | BOOL | CryptDecodeObjectEx(_In_ DWORD dwCertEncodingType, _In_ LPCWSTR pszStructType, _In_ const BYTE *pbEncoded, _In_ DWORD dwFlags, _In_ PCRYPT_DECODE_PARA pDecodePara, _Out_ void *pvStructInfo, _Out_ PCCERT_CONTEXT *pcbStructInfo)                                                                                                                                       |
| ttuituittti  | BOOL | CryptDecryptAndVerifyMessage(_In_ PCRYPT_DECRYPT_MESSAGE_BLOCK *pMsg, _In_ PCRYPT_VERIFY_MESSAGE_PARA pVerifyPara, _In_ DWORD dwSignerIndex, _In_ const BYTE *pbEncryptedBlob, _In_ DWORD dwFlags, _Out_opt_ BYTE *pbDecryptedBlob, _Out_opt_ PCCRYPT_DECRYPT_MESSAGE_BLOCK *pcbDecrypted, _Out_opt_ PCCERT_CONTEXT *ppXchgCert, _Out_opt_ PCCERT_CONTEXT *ppSignerCert) |
| ttuittti     | BOOL | CryptDecryptMessage(_In_ PCRYPT_DECRYPT_MESSAGE_BLOCK *pMsg, _In_ const BYTE *pbEncryptedBlob, _Out_opt_ BYTE *pbDecryptedBlob, _Inout_opt_ DWORD *pcbDecrypted, _Out_opt_ PCCERT_CONTEXT *ppXchgCert)                                                                                                                                                                   |
| uiattti      | BOOL | CryptEncodeObject(_In_ DWORD dwCertEncodingType, _In_ LPCWSTR lpszStructType, _In_ const void *pvStructInfo, _Out_ BYTE *pbEncoded, _Out_ DWORD *pcbEncoded)                                                                                                                                                                                                             |
| uiatuittti   | BOOL | CryptEncodeObjectEx(_In_ DWORD dwCertEncodingType, _In_ LPCWSTR pszStructType, _In_ const void *pvStructInfo, _In_ DWORD dwFlags, _In_ PCRYPT_ENCODE_PARA pEncodePara, _Out_ void *pvEncoded, _Inout_ DWORD *pcbEncoded)                                                                                                                                                 |
| tuittuittti  | BOOL | CryptEncryptMessage(_In_ PCRYPT_ENCRYPT_MESSAGE_BLOCK *pMsg, _In_ DWORD cRecipientCert, _In_ const BINARY_NAME *rgpRecipientCert, _In_ const BYTE *pbToBeEncrypted, _In_ DWORD cbToBeEncrypted, _Out_ void *pbEncryptedBlob, _Inout_ DWORD *pcbEncrypted)                                                                                                                |
| tuiuiwttti   | BOOL | CryptEnumKeyIdentifierProperties(_In_ PCRYPT_HASH_BLOB *pKeyId, _In_ DWORD dwPropId, _In_ DWORD dwFlags, _In_ const PWSTR pszComputerName, _In_ void *pvArg, _In_ PFN_CRYPT_ENUM_KEY_IDENTIFIER_PROPERTIES pfnEnum)                                                                                                                                                      |
| uiaauiitti   | BOOL | CryptEnumOIDFunction(_In_ DWORD dwCertEncodingType, _In_ LPCWSTR pszFuncName, _In_ DWORD dwFlags, _In_ void *pvArg, _In_ PFN_CRYPT_ENUM_OID_FUNC pfnEnum)                                                                                                                                                                                                                |

|               |                  |                                                                                                                                                                                                                                                                                         |
|---------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuitti       | BOOL             | CryptEnumOIDInfo(_In_ DWORD dwFlags, _In_ void *pPFN_CRYPT_ENUM_OID_INFO)                                                                                                                                                                                                               |
| tuiauitti     | BOOL             | CryptExportPKCS8(_In_ HCRYPTPROV_OR_NCRYPT_KEY_HANDLE hCryptProvOrNCryptKey, _In_ DWORD dwKeySpec, _In_ LPCWSTR pszKeySpec, _In_ DWORD dwFlags, _In_opt_ void *pvAuxInfo, _Out_opt_ BYTE *pbPrivateKey, _Out_opt_ BYTE *pcbPrivateKeyBlob)                                              |
| tuiuiti       | BOOL             | CryptExportPublicKeyInfo(_In_ HCRYPTPROV_OR_NCRYPT_KEY_HANDLE hCryptProvOrNCryptKey, _In_ DWORD dwCertEncodingType, _In_ PCERT_PUBLIC_KEY_INFO pCertPublicKeyInfo, _Out_ PCERT_PUBLIC_KEY_INFO *pcbInfo)                                                                                |
| tuiuiauitti   | BOOL             | CryptExportPublicKeyInfoEx(_In_ HCRYPTPROV_OR_NCRYPT_KEY_HANDLE hCryptProvOrNCryptKey, _In_ DWORD dwCertEncodingType, _In_ PCERT_PUBLIC_KEY_INFO pCertPublicKeyInfo, _In_ DWORD dwCertEncodingType, _In_ PCERT_PUBLIC_KEY_INFO *pvAuxInfo, _Out_ PCERT_PUBLIC_KEY_INFO *pcbInfo)        |
| tuiauittui    | BOOL             | CryptExportPublicKeyInfoFromBCRYPT_KEY_HANDLE(_In_ BCRYPT_KEY_HANDLE hBCryptKey, _In_ DWORD dwCertEncodingType, _In_opt_ PCERT_PUBLIC_KEY_INFO pCertPublicKeyInfo, _In_ DWORD dwCertEncodingType, _Out_opt_ PCERT_PUBLIC_KEY_INFO *pvAuxInfo, _Out_opt_ PCERT_PUBLIC_KEY_INFO *pcbInfo) |
| tuiti         | BOOL             | CryptFindCertificateKeyProvId(PCCERT_CONTEXT pCert, _In_ void *pvReserved)                                                                                                                                                                                                              |
| ww            | LPCWSTR          | CryptFindLocalizedName(_In_ LPCWSTR pszCryptName)                                                                                                                                                                                                                                       |
| uituit        | PCCRYPT_OID_INFO | CryptFindOIDInfo(_In_ DWORD dwGroup, _In_ void *pvKey, _In_ DWORD dwGroup)                                                                                                                                                                                                              |
| uiuiuitatuiti | BOOL             | CryptFormatObject(_In_ DWORD dwFormatType, _In_ DWORD dwFormatStrType, _In_ void *pStruct, _In_ LPCSTR lpszStructType, _In_ void *pvAuxInfo, _In_ DWORD cbEncoded, _Out_ DWORD *pcbFormat)                                                                                              |
| tuii          | BOOL             | CryptFreeOIDFunctionAddress(HCRYPTOIDFUNCADDR hFuncSet, _In_ DWORD dwFlags)                                                                                                                                                                                                             |
| tuiwti        | BOOL             | CryptGetDefaultOIDDllList(_In_ HCRYPTOIDFUNCADDR hFuncSet, _In_ DWORD dwEncodingType, _Out_ LPWSTR ppszDllList, _Inout_ void *pvReserved)                                                                                                                                               |
|               |                  | CryptGetDefaultOIDFunctionAddress(HCRYPTOIDFUNCADDR hFuncSet, _In_ DWORD dwFlags)                                                                                                                                                                                                       |



|             |                  |                                                                                                                                                   |
|-------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
|             |                  | PCERT_PUBLIC_KEY_INFO p<br>*pbComputedHash, _Inout_ DW                                                                                            |
| tuituitti   | BOOL             | CryptHashToBeSigned(_In_ HC<br>hCryptProv, _In_ DWORD dwC<br>const BYTE *pbEncoded, _In_ I<br>BYTE *pbComputedHash, _Ino<br>*pcbComputedHash)     |
| uiuitti     | BOOL             | CryptImportPKCS8(_In_<br>CRYPT_PKCS8_IMPORT_PAR<br>sPrivateKeyAndParams, _In_ D<br>_Out_opt_ HCRYPTPROV *phC<br>*pvAuxInfo)                       |
| tuitti      | BOOL             | CryptImportPublicKeyInfo(_In_<br>hCryptProv, _In_ DWORD dwC<br>PCERT_PUBLIC_KEY_INFO p<br>*phKey)                                                 |
| tuituiutti  | BOOL             | CryptImportPublicKeyInfoEx(_I<br>hCryptProv, _In_ DWORD dwC<br>PCERT_PUBLIC_KEY_INFO p<br>aiKeyAlg, _In_ DWORD dwFla<br>_Out_ HCRYPTKEY *phKey)   |
| uituitti    | BOOL             | CryptImportPublicKeyInfoEx2(_<br>dwCertEncodingType, _In_ PCE<br>pInfo, _In_ DWORD dwFlags, _<br>_Out_ BCRYPT_KEY_HANDLE                          |
| auit        | HCRYPTOIDFUNCSET | CryptInitOIDFunctionSet(_In_ I<br>_In_ DWORD dwFlags)                                                                                             |
| tuituitti   | BOOL             | CryptInstallDefaultContext(_In_<br>hCryptProv, _In_ DWORD dwD<br>*pvDefaultPara, _In_ DWORD c<br>*pvReserved, _Out_ HCRYPTD<br>*phDefaultContext) |
| tuiaiuiuiii | BOOL             | CryptInstallOIDFunctionAddres<br>hModule, _In_ DWORD dwEnc<br>pszFuncName, _In_ DWORD c<br>CRYPT_OID_FUNC_ENTRY r<br>DWORD dwFlags)               |
| uit         | LPVOID           | CryptMemAlloc(_In_ ULONG c                                                                                                                        |
| ti          | VOID             | CryptMemFree(_In_ LPVOID p                                                                                                                        |
| tuit        | LPVOID           | CryptMemRealloc(_In_ LPVOID                                                                                                                       |
| uiuiuitaiui | DWORD            | CryptMsgCalculateEncodedLen<br>dwMsgEncodingType, _In_ DW<br>DWORD dwMsgType, _In_ con<br>_In_opt_ LPSTR pszInnerConte<br>cbData)                 |
| ti          | BOOL             | CryptMsgClose(_In_ HCRYPTM                                                                                                                        |

|             |           |                                                                                                                                                                                    |
|-------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiti     | BOOL      | CryptMsgControl(_In_ HCRYPT<br>DWORD dwFlags, _In_ DWOR<br>void *pvCtrlPara)                                                                                                       |
| tuiuiti     | BOOL      | CryptMsgCountersign(_Inout_ P<br>_In_ DWORD dwIndex, _In_ D<br>_In_ PCMSG_SIGNER_ENCOI<br>rgCountersigners)                                                                        |
| uituiuitti  | BOOL      | CryptMsgCountersignEncoded(<br>dwEncodingType, _In_ PBYTE<br>DWORD cbSignerInfo, _In_ DV<br>_In_ PCMSG_SIGNER_ENCOI<br>rgCountersigners, _Out_ PBYTE<br>_Inout_ PDWORD pcbCounters |
| tt          | HCRYPTMSG | CryptMsgDuplicate(_In_ HCRY                                                                                                                                                        |
| uittuitti   | BOOL      | CryptMsgEncodeAndSignCTL(<br>dwMsgEncodingType, _In_ PCT<br>PCMSG_SIGNED_ENCODE_I<br>DWORD dwFlags, _Out_ BYTE<br>DWORD *pcbEncoded)                                               |
| tuituitti   | BOOL      | CryptMsgGetAndVerifySigner(<br>hCryptMsg, _In_ DWORD cSig<br>HCERTSTORE *rghSignerStore<br>_Out_opt_ PCCERT_CONTEXT<br>DWORD *pdwSignerIndex)                                      |
| tuiuitti    | BOOL      | CryptMsgGetParam(_In_ HCRY<br>DWORD dwParamType, _In_ D<br>void *pvData, _Inout_ DWORD                                                                                             |
| uiuiuitttt  | HCRYPTMSG | CryptMsgOpenToDecode(_In_ I<br>dwMsgEncodingType, _In_ DW<br>DWORD dwMsgType, _In_ HC<br>hCryptProv, _In_ PCERT_INFO<br>PCMSG_STREAM_INFO pStre                                    |
| uiuiuitatt  | HCRYPTMSG | CryptMsgOpenToEncode(_In_ I<br>dwMsgEncodingType, _In_ DW<br>DWORD dwMsgType, _In_ con<br>_In_opt_ LPSTR pszInnerConte<br>PCMSG_STREAM_INFO pStre                                  |
| uituituitti | BOOL      | CryptMsgSignCTL(_In_ DWOR<br>_In_ BYTE *pbCtlContent, _In_<br>_In_ PCMSG_SIGNED_ENCOI<br>DWORD dwFlags, _Out_ BYTE<br>DWORD *pcbEncoded)                                           |
| ttuiii      | BOOL      | CryptMsgUpdate(_In_ HCRYPT<br>const BYTE *pbData, _In_ DW<br>fFinal)                                                                                                               |
|             |           | CryptMsgVerifyCountersignatur                                                                                                                                                      |





|             |      |                                                                                                                 |
|-------------|------|-----------------------------------------------------------------------------------------------------------------|
| ti          | BOOL | *psNewProv)                                                                                                     |
| ttti        | BOOL | CryptSIPCreateIndirectData(_In_ *pSubjectInfo, _Inout_ DWORD SIP_INDIRECT_DATA *pIndire                         |
| ttuitti     | BOOL | CryptSIPGetSignedDataMsg(_In_ *pSubjectInfo, _Out_ DWORD DWORD dwIndex, _Inout_ DW _Out_ BYTE *pbSignedDataMs   |
| tuiti       | BOOL | CryptSIPLoad(_In_ const GUID dwFlags, _Inout_ SIP_DISPATC                                                       |
| tuituiti    | BOOL | CryptSIPPutSignedDataMsg(_In_ *pSubjectInfo, _In_ DWORD dw DWORD *pdwIndex, _In_ DW _In_ BYTE *pbSignedDataMsg) |
| ti          | BOOL | CryptSIPRemoveProvider(_In_ (                                                                                   |
| tuii        | BOOL | CryptSIPRemoveSignedDataMs SIP_SUBJECTINFO *pSubjectI dwIndex)                                                  |
| wtti        | BOOL | CryptSIPRetrieveSubjectGuid(_ _In_opt_ HANDLE hFileIn, _Ou                                                      |
| wtti        | BOOL | CryptSIPRetrieveSubjectGuidPe LPCWSTR FileName, _In_opt_ GUID *pgSubject)                                       |
| titi        | BOOL | CryptSIPVerifyIndirectData(_In_ *pSubjectInfo, _In_ SIP_INDIR *pIndirectData)                                   |
| suiuittiti  | BOOL | CryptStringToBinary(_In_ LPCT DWORD cchString, _In_ DWOI *pbBinary, _Inout_ DWORD *p *pdwSkip, _Out_ DWORD *pdv |
| aiuiuittiti | BOOL | CryptStringToBinaryA(_In_ LPC DWORD cchString, _In_ DWOI *pbBinary, _Inout_ DWORD *p *pdwSkip, _Out_ DWORD *pdv |
| wuiuittiti  | BOOL | CryptStringToBinaryW(_In_ LP DWORD cchString, _In_ DWOI *pbBinary, _Inout_ DWORD *p *pdwSkip, _Out_ DWORD *pdv  |
| tuiti       | BOOL | CryptUninstallDefaultContext(_ HCRYPTDEFAULTCONTEXT DWORD dwFlags, _In_ void *p                                 |
| ttttuiti    | BOOL | CryptUnprotectData(_In_ DATA _Out_opt_ LPWSTR *ppszData DATA_BLOB *pOptionalEntrop pvReserved, _In_opt_         |

|             |      |                                                                                                                                                                                                                                       |
|-------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |      | CRYPTPROTECT_PROMPTST<br>_In_ DWORD dwFlags, _Out_ I                                                                                                                                                                                  |
| tuiiii      | BOOL | CryptUnprotectMemory(_Inout_ DWORD cbData, _In_ DWORD                                                                                                                                                                                 |
| uiaai       | BOOL | CryptUnregisterDefaultOIDFunc<br>dwEncodingType, _In_ LPCSTR<br>LPCWSTR pwszDll)                                                                                                                                                      |
| uiaai       | BOOL | CryptUnregisterOIDFunction(_I<br>dwEncodingType, _In_ LPCSTR<br>LPCSTR pszOID)                                                                                                                                                        |
| ti          | BOOL | CryptUnregisterOIDInfo(_In_ P<br>pInfo)                                                                                                                                                                                               |
| twuitti     | BOOL | CryptUpdateProtectedState(_In_ LPCWSTR pwszOldPassword, _<br>_Out_ DWORD *pdwSuccessC<br>*pdwFailureCount)                                                                                                                            |
| tuituiti    | BOOL | CryptVerifyCertificateSignature(<br>HCRYPTPROV_LEGACY hCry<br>dwCertEncodingType, _In_ BYT<br>DWORD cbEncoded, _In_ PCE<br>pPublicKey)                                                                                                |
| tuituituiti | BOOL | CryptVerifyCertificateSignature(<br>HCRYPTPROV_LEGACY hCry<br>dwCertEncodingType, _In_ DW<br>void *pvSubject, _In_ DWORD<br>*pvIssuer, _In_ DWORD dwFla<br>*pvExtra)                                                                  |
| ttuituiti   | BOOL | CryptVerifyDetachedMessageHa<br>PCRYPT_HASH_MESSAGE_P<br>BYTE *pbDetachedHashBlob, _<br>cbDetachedHashBlob, _In_ DW<br>const BYTE *rgpbToBeHashed[<br>rgcbToBeHashed[], _Out_ BYT<br>_Inout_ DWORD *pcbCompute                        |
| tuituituiti | BOOL | CryptVerifyDetachedMessageSi<br>PCRYPT_VERIFY_MESSAGE<br>DWORD dwSignerIndex, _In_ c<br>*pbDetachedSignBlob, _In_ DW<br>cbDetachedSignBlob, _In_ DW<br>const BYTE *rgpbToBeSigned[<br>rgcbToBeSigned[], _Out_opt_ P<br>*ppSignerCert) |
| ttuittti    | BOOL | CryptVerifyMessageHash(_In_ PCRYPT_HASH_MESSAGE_P<br>BYTE *pbHashedBlob, _In_ DV<br>_Out_ BYTE *pbToBeHashed, _<br>*pcbToBeHashed, _Out_opt_ B                                                                                        |

|              |            |                                                                                                                                                                                                    |
|--------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |            | _Inout_opt_ DWORD *pcbCom                                                                                                                                                                          |
| tuituitti    | BOOL       | CryptVerifyMessageSignature(<br>PCRYPT_VERIFY_MESSAGE<br>DWORD dwSignerIndex, _In_ c<br>*pbSignedBlob, _In_ DWORD c<br>BYTE *pbDecoded, _Inout_ DV<br>_Out_opt_ PCCERT_CONTEXT                     |
| ttuitti      | BOOL       | CryptVerifyMessageSignatureW<br>PCRYPT_KEY_VERIFY_MES<br>_In_ PCERT_PUBLIC_KEY_IN<br>const BYTE *pbSignedBlob, _In<br>_Out_ BYTE *pbDecoded, _Ino<br>*pcbDecoded)                                  |
| ucuiuiuittti | BOOL       | CryptVerifyTimeStampSignature<br>pbTSContentInfo, DWORD cbT<br>const DWORD pbData, DWOR<br>HCERTSTORE hAdditionalStor<br>PCRYPT_TIMESTAMP_CONT<br>_Out_opt_ PCCERT_CONTEXT<br>HCERTSTORE *phStore) |
| ttwuii       | BOOL       | PFXExportCertStore(_In_ HCEL<br>CRYPT_DATA_BLOB *pPFX,<br>szPassword, _In_ DWORD dwF                                                                                                               |
| ttwtuii      | BOOL       | PFXExportCertStoreEx(_In_ HC<br>_Inout_ CRYPT_DATA_BLOB<br>szPassword, _In_ void *pvPara,                                                                                                          |
| twuit        | HCERTSTORE | PFXImportCertStore(_In_ CRY<br>_In_ LPCWSTR szPassword, _I                                                                                                                                         |
| ti           | BOOL       | PFXIsPFXBlob(_In_ CRYPT_D                                                                                                                                                                          |
| twuii        | BOOL       | PFXVerifyPassword(_In_ CRY<br>_In_ LPCWSTR szPassword, _I                                                                                                                                          |

# Gdi32.dll

|            |        |                                                                                                                                                                                       |
|------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti         | int    | AbortDoc(_In_ HDC hdc)                                                                                                                                                                |
| ti         | BOOL   | AbortPath(_In_ HDC hdc)                                                                                                                                                               |
| tuiitt     | HANDLE | AddFontMemResourceEx(_In_ PVOID pbFont, _In_ DWORD cbFont, _In_ PVOID pdv, _In_ DWORD *pcFonts)                                                                                       |
| si         | int    | AddFontResource(_In_ LPCTSTR lpszFilename)                                                                                                                                            |
| ai         | int    | AddFontResourceA(_In_ LPCSTR lpszFilename)                                                                                                                                            |
| suiti      | int    | AddFontResourceEx(_In_ LPCTSTR lpszFilename, _In_ DWORD fl, _In_ PVOID pdv)                                                                                                           |
| auti       | int    | AddFontResourceExA(_In_ LPCSTR lpszFilename, _In_ DWORD fl, _In_ PVOID pdv)                                                                                                           |
| wuiti      | int    | AddFontResourceExW(_In_ LPCWSTR lpszFilename, _In_ DWORD fl, _In_ PVOID pdv)                                                                                                          |
| wi         | int    | AddFontResourceW(_In_ LPCWSTR lpszFilename)                                                                                                                                           |
| tiiuiffi   | BOOL   | AngleArc(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ DWORD dwRadius, _In_ FLOAT eStartAngle, _In_ FLOAT eSweepAngle)                                                                   |
| tuiuiti    | BOOL   | AnimatePalette(_In_ HPALETTE hpal, _In_ UINT iStartIndex, _In_ UINT cEntries, _In_ const PALETTEENTRY *ppe)                                                                           |
| tiiiiiii   | BOOL   | Arc(_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect, _In_ int nXStartArc, _In_ int nYStartArc, _In_ int nXEndArc, _In_ int nYEndArc)   |
| tiiiiiii   | BOOL   | ArcTo(_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect, _In_ int nXRadial1, _In_ int nYRadial1, _In_ int nXRadial2, _In_ int nYRadial2) |
| ti         | BOOL   | BeginPath(_In_ HDC hdc)                                                                                                                                                               |
| tiiitiuiii | BOOL   | BitBlt(_In_ HDC hdcDest, _In_ int nXDest, _In_ int nYDest, _In_ int nWidth, _In_ int nHeight, _In_ HDC hdcSrc, _In_ int nXSrc, _In_ int nYSrc, _In_ DWORD dwRop)                      |
| ti         | BOOL   | CancelDC(_In_ HDC hdc)                                                                                                                                                                |
| tttiii     | BOOL   | CheckColorsInGamut(HDC hdc, LPVOID lpRGBTriples, LPVOID lpBuffer, UINT nCount)                                                                                                        |
| tii        | int    | ChoosePixelFormat(HDC hdc, const PIXELFORMATDESCRIPTOR *ppfd)                                                                                                                         |

|            |              |                                                                                                                                                                                               |
|------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiiiiitiii | BOOL         | <i>Chord</i> (_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect, _In_ int nXRadial1, _In_ int nYRadial1, _In_ int nXRadial2, _In_ int nYRadial2) |
| tt         | HENHMETAFILE | <i>CloseEnhMetaFile</i> (_In_ HDC hdc)                                                                                                                                                        |
| ti         | BOOL         | <i>CloseFigure</i> (_In_ HDC hdc)                                                                                                                                                             |
| tt         | HMETAFILE    | <i>CloseMetaFile</i> (_In_ HDC hdc)                                                                                                                                                           |
| ttuiiii    | BOOL         | <i>ColorCorrectPalette</i> (HDC hdc, HPALETTE hPalette, DWORD dwFirstEntry, DWORD dwNumOfEntries)                                                                                             |
| ttuii      | BOOL         | <i>ColorMatchToTarget</i> (HDC hdc, HDC hdcTarget, DWORD uiAction)                                                                                                                            |
| tttii      | int          | <i>CombineRgn</i> (_In_ HRGN hrgnDest, _In_ HRGN hrgnSrc1, _In_ HRGN hrgnSrc2, _In_ int fnCombineMode)                                                                                        |
| ttti       | BOOL         | <i>CombineTransform</i> (_Out_ LPXFORM lpxformResult, _In_ const XFORM *lpxform1, _In_ const XFORM *lpxform2)                                                                                 |
| tst        | HENHMETAFILE | <i>CopyEnhMetaFile</i> (_In_ HENHMETAFILE hemfSrc, _In_ LPCTSTR lpszFile)                                                                                                                     |
| tat        | HENHMETAFILE | <i>CopyEnhMetaFileA</i> (_In_ HENHMETAFILE hemfSrc, _In_ LPCSTR lpszFile)                                                                                                                     |
| twt        | HENHMETAFILE | <i>CopyEnhMetaFileW</i> (_In_ HENHMETAFILE hemfSrc, _In_ LPCWSTR lpszFile)                                                                                                                    |
| tst        | HMETAFILE    | <i>CopyMetaFile</i> (_In_ HMETAFILE hmfSrc, _In_ LPCTSTR lpszFile)                                                                                                                            |
| tat        | HMETAFILE    | <i>CopyMetaFileA</i> (_In_ HMETAFILE hmfSrc, _In_ LPCSTR lpszFile)                                                                                                                            |
| twt        | HMETAFILE    | <i>CopyMetaFileW</i> (_In_ HMETAFILE hmfSrc, _In_ LPCWSTR lpszFile)                                                                                                                           |
| iiuiiitt   | HBITMAP      | <i>CreateBitmap</i> (_In_ int nWidth, _In_ int nHeight, _In_ UINT cPlanes, _In_ UINT cBitsPerPel, _In_ const VOID *lpvBits)                                                                   |
| tt         | HBITMAP      | <i>CreateBitmapIndirect</i> (_In_ const BITMAP *lpbm)                                                                                                                                         |
| tt         | HBRUSH       | <i>CreateBrushIndirect</i> (_In_ const LOGBRUSH *lpfb)                                                                                                                                        |
| tt         | HCOLORSPACE  | <i>CreateColorSpace</i> (LPLOGCOLORSPACE lpLogColorSpace)                                                                                                                                     |
| tt         | HCOLORSPACE  | <i>CreateColorSpaceA</i> (LPLOGCOLORSPACE lpLogColorSpace)                                                                                                                                    |
| tt         | HCOLORSPACE  | <i>CreateColorSpaceW</i> (LPLOGCOLORSPACE lpLogColorSpace)                                                                                                                                    |
| tiit       | HBITMAP      | <i>CreateCompatibleBitmap</i> (_In_ HDC hdc, _In_ int nWidth, _In_ int nHeight)                                                                                                               |

|                        |         |                                                                                                                                                                                                                                                                                                                                                      |
|------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tt                     | HDC     | CreateCompatibleDC(_In_ HDC hdc)                                                                                                                                                                                                                                                                                                                     |
| ssstt                  | HDC     | CreateDC(LPCTSTR lpszDriver, _In_ LPCTSTR lpszDevice, LPCTSTR lpszOutput, _In_ const DEVMODE *lpInitData)                                                                                                                                                                                                                                            |
| aaatt                  | HDC     | CreateDCA(LPCSTR lpszDriver, _In_ LPCSTR lpszDevice, LPCSTR lpszOutput, _In_ const DEVMODE *lpInitData)                                                                                                                                                                                                                                              |
| wwwtt                  | HDC     | CreateDCW(LPCWSTR lpszDriver, _In_ LPCWSTR lpszDevice, LPCWSTR lpszOutput, _In_ const DEVMODE *lpInitData)                                                                                                                                                                                                                                           |
| ttuittuit              | HBITMAP | CreateDIBitmap(_In_ HDC hdc, _In_ const BITMAPINFOHEADER *lpbmih, _In_ DWORD fdwInit, _In_ const VOID *lpbInit, _In_ const BITMAPINFO *lpbmi, _In_ UINT fuUsage)                                                                                                                                                                                     |
| tuit                   | HBRUSH  | CreateDIBPatternBrush(_In_ HGLOBAL hgldbDIBPacked, _In_ UINT fuColorSpec)                                                                                                                                                                                                                                                                            |
| tuit                   | HBRUSH  | CreateDIBPatternBrushPt(_In_ const VOID *lpPackedDIB, _In_ UINT iUsage)                                                                                                                                                                                                                                                                              |
| ttuittuit              | HBITMAP | CreateDIBSection(_In_ HDC hdc, _In_ const BITMAPINFO *pbmi, _In_ UINT iUsage, _Out_ VOID **ppvBits, _In_ HANDLE hSection, _In_ DWORD dwOffset)                                                                                                                                                                                                       |
| tiit                   | HBITMAP | CreateDiscardableBitmap(_In_ HDC hdc, _In_ int nWidth, _In_ int nHeight)                                                                                                                                                                                                                                                                             |
| iiiiit                 | HRGN    | CreateEllipticRgn(_In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                                                                                                                                                                                                                                                  |
| tt                     | HRGN    | CreateEllipticRgnIndirect(_In_ const RECT *lprc)                                                                                                                                                                                                                                                                                                     |
| tstst                  | HDC     | CreateEnhMetaFile(_In_ HDC hdcRef, _In_ LPCTSTR lpFilename, _In_ const RECT *lpRect, _In_ LPCTSTR lpDescription)                                                                                                                                                                                                                                     |
| tatat                  | HDC     | CreateEnhMetaFileA(_In_ HDC hdcRef, _In_ LPCSTR lpFilename, _In_ const RECT *lpRect, _In_ LPCSTR lpDescription)                                                                                                                                                                                                                                      |
| twtwt                  | HDC     | CreateEnhMetaFileW(_In_ HDC hdcRef, _In_ LPCWSTR lpFilename, _In_ const RECT *lpRect, _In_ LPCWSTR lpDescription)                                                                                                                                                                                                                                    |
| iiiiiuuuuuuuuuuuuuuist | HFONT   | CreateFont(_In_ int nHeight, _In_ int nWidth, _In_ int nEscapement, _In_ int nOrientation, _In_ int fnWeight, _In_ DWORD fdwItalic, _In_ DWORD fdwUnderline, _In_ DWORD fdwStrikeOut, _In_ DWORD fdwCharSet, _In_ DWORD fdwOutputPrecision, _In_ DWORD fdwClipPrecision, _In_ DWORD fdwQuality, _In_ DWORD fdwPitchAndFamily, _In_ LPCTSTR lpszFace) |
|                        |         | CreateFontA(_In_ int nHeight, _In_ int nWidth, _In_ int                                                                                                                                                                                                                                                                                              |



|         |        |                                                                                                                                                                   |
|---------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiit    | HRGN   | CreatePolygonRgn(_In_ const POINT *lppt, _In_ int cPoints, _In_ int fnPolyFillMode)                                                                               |
| ttiiit  | HRGN   | CreatePolyPolygonRgn(_In_ const POINT *lppt, _In_ const INT *lpPolyCounts, _In_ int nCount, _In_ int fnPolyFillMode)                                              |
| iiiiit  | HRGN   | CreateRectRgn(_In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                                                                   |
| tt      | HRGN   | CreateRectRgnIndirect(_In_ const RECT *lprc)                                                                                                                      |
| iiiiit  | HRGN   | CreateRoundRectRgn(_In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect, _In_ int nWidthEllipse, _In_ int nHeightEllipse)             |
| uissii  | BOOL   | CreateScalableFontResource(_In_ DWORD fdwHidden, _In_ LPCTSTR lpszFontRes, _In_ LPCTSTR lpszFontFile, _In_ LPCTSTR lpszCurrentPath)                               |
| uiaaai  | BOOL   | CreateScalableFontResourceA(_In_ DWORD fdwHidden, _In_ LPCSTR lpszFontRes, _In_ LPCSTR lpszFontFile, _In_ LPCSTR lpszCurrentPath)                                 |
| uiwwwi  | BOOL   | CreateScalableFontResourceW(_In_ DWORD fdwHidden, _In_ LPCWSTR lpszFontRes, _In_ LPCWSTR lpszFontFile, _In_ LPCWSTR lpszCurrentPath)                              |
| uit     | HBRUSH | CreateSolidBrush(_In_ COLORREF crColor)                                                                                                                           |
| tauii   | int    | DDCCIGetCapabilitiesString(__in HANDLE hMonitor, __out LPSTR pszString, __in DWORD dwLength)                                                                      |
| tii     | int    | DDCCIGetCapabilitiesStringLength(__in HANDLE hMonitor, __out DWORD *pdwLength)                                                                                    |
| tii     | int    | DDCCIGetTimingReport(__in HANDLE hMonitor, __out LPMC_TIMING_REPORT pmtr)                                                                                         |
| tuittti | int    | DDCCIGetVCPFeature(__in HANDLE hMonitor, __in DWORD dwVCPCode, __out_opt LPMC_VCP_CODE_TYPE pvct, __out DWORD *pdwCurrentValue, __out_opt DWORD *pdwMaximumValue) |
| ti      | int    | DDCCISaveCurrentSettings(HANDLE hMonitor)                                                                                                                         |
| tuiiii  | int    | DDCCISetVCPFeature(__in HANDLE hMonitor, __in DWORD dwVCPCode, __in DWORD dwNewValue)                                                                             |
| ti      | BOOL   | DeleteColorSpace(HCOLORSPACE hColorSpace)                                                                                                                         |
| ti      | BOOL   | DeleteDC(_In_ HDC hdc)                                                                                                                                            |
| ti      | BOOL   | DeleteEnhMetaFile(_In_ HENHMETAFILE hemf)                                                                                                                         |
| ti      | BOOL   | DeleteMetaFile(_In_ HMETAFILE hmf)                                                                                                                                |
| ti      | BOOL   | DeleteObject(_In_ HGDIOBJ hObject)                                                                                                                                |
|         |        | DescribePixelFormat(HDC hdc, int iPixelFormat, UINT                                                                                                               |

|         |      |                                                                                                                                       |
|---------|------|---------------------------------------------------------------------------------------------------------------------------------------|
| tiuiti  | int  | nBytes, LPPIXELFORMATDESCRIPTOR pPfd)                                                                                                 |
| ti      | int  | DestroyPhysicalMonitorInternal(__in HANDLE hMonitor)                                                                                  |
| ttii    | BOOL | DPrintLP(_In_ HDC hdc, _Inout_ LPPOINT lpPoints, _In_ int nCount)                                                                     |
| tiaai   | int  | DrawEscape(_In_ HDC hdc, _In_ int nEscape, _In_ int cbInput, _In_ LPCSTR lpszInData)                                                  |
| tiiii   | BOOL | Ellipse(_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                               |
| ti      | BOOL | EnableEUDC(_In_ HDC BOOL fEnableEUDC)                                                                                                 |
| ti      | int  | EndDoc(_In_ HDC hdc)                                                                                                                  |
| ti      | int  | EndPage(_In_ HDC hdc)                                                                                                                 |
| ti      | BOOL | EndPath(_In_ HDC hdc)                                                                                                                 |
| tttiti  | BOOL | EnumEnhMetaFile(_In_ HDC hdc, _In_ HENHMETAFILE hemf, _In_ ENHMFENUMPROC lpEnumMetaFunc, _In_ LPVOID lpData, _In_ const RECT *lpRect) |
| tstti   | int  | EnumFontFamilies(_In_ HDC hdc, _In_ LPCTSTR lpszFamily, _In_ FONTENUMPROC lpEnumFontFamProc, _In_ LPARAM lParam)                      |
| tatti   | int  | EnumFontFamiliesA(_In_ HDC hdc, _In_ LPCSTR lpszFamily, _In_ FONTENUMPROC lpEnumFontFamProc, _In_ LPARAM lParam)                      |
| ttttuui | int  | EnumFontFamiliesEx(_In_ HDC hdc, _In_ LPLOGFONT lpLogfont, _In_ FONTENUMPROC lpEnumFontFamExProc, _In_ LPARAM lParam, DWORD dwFlags)  |
| ttttuui | int  | EnumFontFamiliesExA(_In_ HDC hdc, _In_ LPLOGFONT lpLogfont, _In_ FONTENUMPROC lpEnumFontFamExProc, _In_ LPARAM lParam, DWORD dwFlags) |
| ttttuui | int  | EnumFontFamiliesExW(_In_ HDC hdc, _In_ LPLOGFONT lpLogfont, _In_ FONTENUMPROC lpEnumFontFamExProc, _In_ LPARAM lParam, DWORD dwFlags) |
| twtti   | int  | EnumFontFamiliesW(_In_ HDC hdc, _In_ LPCWSTR lpszFamily, _In_ FONTENUMPROC lpEnumFontFamProc, _In_ LPARAM lParam)                     |
| tssti   | int  | EnumFonts(_In_ HDC hdc, _In_ LPCTSTR lpFaceName, _In_ FONTENUMPROC lpFontFunc, _In_ LPARAM lParam)                                    |
| tatti   | int  | EnumFontsA(_In_ HDC hdc, _In_ LPCSTR lpFaceName, _In_ FONTENUMPROC lpFontFunc, _In_ LPARAM                                            |

|              |      |                                                                                                                                                               |
|--------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |      | lParam)                                                                                                                                                       |
| twtti        | int  | EnumFontsW(_In_ HDC hdc, _In_ LPCWSTR lpFaceName, _In_ FONTENUMPROC lpFontFunc, _In_ LPARAM lParam)                                                           |
| ttti         | int  | EnumICMProfiles(HDC hdc, ICMENUMPROC lpEnumICMProfilesFunc, LPARAM lParam)                                                                                    |
| ttti         | int  | EnumICMProfilesA(HDC hdc, ICMENUMPROC lpEnumICMProfilesFunc, LPARAM lParam)                                                                                   |
| ttti         | int  | EnumICMProfilesW(HDC hdc, ICMENUMPROC lpEnumICMProfilesFunc, LPARAM lParam)                                                                                   |
| ttti         | BOOL | EnumMetaFile(_In_ HDC hdc, _In_ HMETAFILE hmf, _In_ MFENUMPROC lpMetaFunc, _In_ LPARAM lParam)                                                                |
| titti        | int  | EnumObjects(_In_ HDC hdc, _In_ int nObjectType, _In_ GOBJENUMPROC lpObjectFunc, _In_ LPARAM lParam)                                                           |
| titi         | BOOL | EqualRgn(_In_ HRGN hSrcRgn1, _In_ HRGN hSrcRgn2)                                                                                                              |
| tiiati       | int  | Escape(_In_ HDC hdc, _In_ int nEscape, _In_ int cbInput, _In_ LPCSTR lpvInData, _Out_ LPVOID lpvOutData)                                                      |
| tiiii        | int  | ExcludeClipRect(_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                                               |
| uiuituitt    | HPEN | ExtCreatePen(_In_ DWORD dwPenStyle, _In_ DWORD dwWidth, _In_ const LOGBRUSH *lpLb, _In_ DWORD dwStyleCount, _In_ const DWORD *lpStyle)                        |
| tuitt        | HRGN | ExtCreateRegion(_In_ const XFORM *lpXform, _In_ DWORD nCount, _In_ const RGNDATA *lpRgnData)                                                                  |
| tiaiaia      | int  | ExtEscape(_In_ HDC hdc, _In_ int nEscape, _In_ int cbInput, _In_ LPCSTR lpszInData, _In_ int cbOutput, _Out_ LPSTR lpszOutData)                               |
| tiiuiiii     | BOOL | ExtFloodFill(_In_ HDC hdc, _In_ int nXStart, _In_ int nYStart, _In_ COLORREF crColor, _In_ UINT fuFillType)                                                   |
| ttii         | int  | ExtSelectClipRgn(_In_ HDC hdc, _In_ HRGN hrgn, _In_ int fnMode)                                                                                               |
| tiiuitsuitti | BOOL | ExtTextOut(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ UINT fuOptions, _In_ const RECT *lprc, _In_ LPCTSTR lpString, _In_ UINT cbCount, _In_ const INT *lpDx)  |
| tiiuitauitti | BOOL | ExtTextOutA(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ UINT fuOptions, _In_ const RECT *lprc, _In_ LPCSTR lpString, _In_ UINT cbCount, _In_ const INT *lpDx)  |
| tiiuitwuiti  | BOOL | ExtTextOutW(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ UINT fuOptions, _In_ const RECT *lprc, _In_ LPCWSTR lpString, _In_ UINT cbCount, _In_ const INT *lpDx) |

|              |          |                                                                                                                                                                                                                                   |
|--------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti           | BOOL     | FillPath(_In_ HDC hdc)                                                                                                                                                                                                            |
| ttti         | BOOL     | FillRect(_In_ HDC hdc, _In_ HRGN hrgn, _In_ HBRUSH hbr)                                                                                                                                                                           |
| ti           | BOOL     | FlattenPath(_In_ HDC hdc)                                                                                                                                                                                                         |
| tiiuii       | BOOL     | FloodFill(_In_ HDC hdc, _In_ int nXStart, _In_ int nYStart, _In_ COLORREF crFill)                                                                                                                                                 |
| ttiii        | BOOL     | FrameRect(_In_ HDC hdc, _In_ HRGN hrgn, _In_ HBRUSH hbr, _In_ int nWidth, _In_ int nHeight)                                                                                                                                       |
| tiiiiiiiiiii | BOOL     | GdiAlphaBlend(_In_ HDC hdcDest, _In_ int xoriginDest, _In_ int yoriginDest, _In_ int wDest, _In_ int hDest, _In_ HDC hdcSrc, _In_ int xoriginSrc, _In_ int yoriginSrc, _In_ int wSrc, _In_ int hSrc, _In_ BLENDFUNCTION ftn)      |
| tuiti        | BOOL     | GdiComment(_In_ HDC hdc, _In_ UINT cbSize, _In_ const BYTE *lpData)                                                                                                                                                               |
| i            | BOOL     | GdiFlush(void)                                                                                                                                                                                                                    |
| ui           | DWORD    | GdiGetBatchLimit(void)                                                                                                                                                                                                            |
| ttuituiiii   | BOOL     | GdiGradientFill(_In_ HDC hdc, _In_ PTRIVERTEX pVertex, _In_ ULONG dwNumVertex, _In_ PVOID pMesh, _In_ ULONG dwNumMesh, _In_ ULONG dwMode)                                                                                         |
| uiui         | DWORD    | GdiSetBatchLimit(_In_ DWORD dwLimit)                                                                                                                                                                                              |
| tiiiiiiiiiii | BOOL     | GdiTransparentBlt(_In_ HDC hdcDest, _In_ int xoriginDest, _In_ int yoriginDest, _In_ int wDest, _In_ int hDest, _In_ HDC hdcSrc, _In_ int xoriginSrc, _In_ int yoriginSrc, _In_ int wSrc, _In_ int hSrc, _In_ UINT crTransparent) |
| ti           | int      | GetArcDirection(_In_ HDC hdc)                                                                                                                                                                                                     |
| tti          | BOOL     | GetAspectRatioFilterEx(_In_ HDC hdc, _Out_ LPSIZE lpAspectRatio)                                                                                                                                                                  |
| tuitui       | LONG     | GetBitmapBits(_In_ HBITMAP hbmp, _In_ LONG cbBuffer, _Out_ LPVOID lpvBits)                                                                                                                                                        |
| tti          | BOOL     | GetBitmapDimensionEx(_In_ HBITMAP hBitmap, _Out_ LPSIZE lpDimension)                                                                                                                                                              |
| tui          | COLORREF | GetBkColor(_In_ HDC hdc)                                                                                                                                                                                                          |
| ti           | int      | GetBkMode(_In_ HDC hdc)                                                                                                                                                                                                           |
| ttuiui       | UINT     | GetBoundsRect(_In_ HDC hdc, _Out_ LPRECT lprcBounds, _In_ UINT flags)                                                                                                                                                             |
| tti          | BOOL     | GetBrushOrgEx(_In_ HDC hdc, _Out_ LPPOINT lppt)                                                                                                                                                                                   |
| tuiuiti      | BOOL     | GetCharABCWidths(_In_ HDC hdc, _In_ UINT uFirstChar, _In_ UINT uLastChar, _Out_ LPABC lpabc)                                                                                                                                      |

|           |       |                                                                                                                                                        |
|-----------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiti   | BOOL  | GetCharABCWidthsA(_In_ HDC hdc, _In_ UINT uFirstChar, _In_ UINT uLastChar, _Out_ LPABC lpabc)                                                          |
| tuiuiti   | BOOL  | GetCharABCWidthsFloat(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPABCFLOAT lpABCF)                                                |
| tuiuiti   | BOOL  | GetCharABCWidthsFloatA(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPABCFLOAT lpABCF)                                               |
| tuiuiti   | BOOL  | GetCharABCWidthsFloatW(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPABCFLOAT lpABCF)                                               |
| tuiuiti   | BOOL  | GetCharABCWidthsI(_In_ HDC hdc, _In_ UINT giFirst, _In_ UINT cgi, _In_ LPWORD pgi, _Out_ LPABC lpabc)                                                  |
| tuiuiti   | BOOL  | GetCharABCWidthsW(_In_ HDC hdc, _In_ UINT uFirstChar, _In_ UINT uLastChar, _Out_ LPABC lpabc)                                                          |
| tsiituiui | DWORD | GetCharacterPlacement(_In_ HDC hdc, _In_ LPCTSTR lpString, _In_ int nCount, _In_ int nMaxExtent, _Inout_ LPGCP_RESULTS lpResults, _In_ DWORD dwFlags)  |
| taiituiui | DWORD | GetCharacterPlacementA(_In_ HDC hdc, _In_ LPCSTR lpString, _In_ int nCount, _In_ int nMaxExtent, _Inout_ LPGCP_RESULTS lpResults, _In_ DWORD dwFlags)  |
| twiituiui | DWORD | GetCharacterPlacementW(_In_ HDC hdc, _In_ LPCWSTR lpString, _In_ int nCount, _In_ int nMaxExtent, _Inout_ LPGCP_RESULTS lpResults, _In_ DWORD dwFlags) |
| tuiuiti   | BOOL  | GetCharWidth(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPINT lpBuffer)                                                            |
| tuiuiti   | BOOL  | GetCharWidth32(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPINT lpBuffer)                                                          |
| tuiuiti   | BOOL  | GetCharWidth32A(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPINT lpBuffer)                                                         |
| tuiuiti   | BOOL  | GetCharWidth32W(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPINT lpBuffer)                                                         |
| tuiuiti   | BOOL  | GetCharWidthA(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPINT lpBuffer)                                                           |
| tuiuiti   | BOOL  | GetCharWidthFloat(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ PFLOAT pxBuffer)                                                      |
| tuiuiti   | BOOL  | GetCharWidthFloatA(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ PFLOAT pxBuffer)                                                     |
| tuiuiti   | BOOL  | GetCharWidthFloatW(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ PFLOAT pxBuffer)                                                     |
|           |       |                                                                                                                                                        |

|             |              |                                                                                                                                                          |
|-------------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiti     | BOOL         | GetCharWidth(_In_ HDC hdc, _In_ UINT giFirst, _In_ UINT cgi, _In_ LPWORD pgi, _Out_ LPINT lpBuffer)                                                      |
| tuiuiti     | BOOL         | GetCharWidthW(_In_ HDC hdc, _In_ UINT iFirstChar, _In_ UINT iLastChar, _Out_ LPINT lpBuffer)                                                             |
| tti         | int          | GetClipBox(_In_ HDC hdc, _Out_ LPRECT lprc)                                                                                                              |
| tti         | int          | GetClipRgn(_In_ HDC hdc, _In_ HRGN hrgn)                                                                                                                 |
| tti         | BOOL         | GetColorAdjustment(_In_ HDC hdc, _Out_ LPCOLORADJUSTMENT lpca)                                                                                           |
| tt          | HCOLORSPACE  | GetColorSpace(HDC hdc)                                                                                                                                   |
| tuit        | HGDIOBJ      | GetCurrentObject(_In_ HDC hdc, _In_ UINT uObjectType)                                                                                                    |
| tti         | BOOL         | GetCurrentPositionEx(_In_ HDC hdc, _Out_ LPPOINT lpPoint)                                                                                                |
| tui         | COLORREF     | GetDCBrushColor(_In_ HDC hdc)                                                                                                                            |
| tti         | BOOL         | GetDCOrgEx(_In_ HDC hdc, _Out_ LPPOINT lpPoint)                                                                                                          |
| tui         | COLORREF     | GetDCPenColor(_In_ HDC hdc)                                                                                                                              |
| tii         | int          | GetDeviceCaps(_In_ HDC hdc, _In_ int nIndex)                                                                                                             |
| tti         | BOOL         | GetDeviceGammaRamp(HDC hdc, LPVOID lpRamp)                                                                                                               |
| tuiuitui    | UINT         | GetDIBColorTable(_In_ HDC hdc, _In_ UINT uStartIndex, _In_ UINT cEntries, _Out_ RGBQUAD *pColors)                                                        |
| ttuiuittiii | int          | GetDIBits(_In_ HDC hdc, _In_ HBITMAP hbm, _In_ UINT uStartScan, _In_ UINT cScanLines, _Out_ LPVOID lpvBits, _Inout_ LPBITMAPINFO lpbi, _In_ UINT uUsage) |
| st          | HENHMETAFILE | GetEnhMetaFile(_In_ LPCTSTR lpszMetaFile)                                                                                                                |
| at          | HENHMETAFILE | GetEnhMetaFileA(_In_ LPCSTR lpszMetaFile)                                                                                                                |
| tuitui      | UINT         | GetEnhMetaFileBits(_In_ HENHMETAFILE hemf, _In_ UINT cbBuffer, _Out_ LPBYTE lpbBuffer)                                                                   |
| tuisui      | UINT         | GetEnhMetaFileDescription(_In_ HENHMETAFILE hemf, _In_ UINT cchBuffer, _Out_ LPTSTR lpszDescription)                                                     |
| tuiiui      | UINT         | GetEnhMetaFileDescriptionA(_In_ HENHMETAFILE hemf, _In_ UINT cchBuffer, _Out_ LPSTR lpszDescription)                                                     |
| tuiwui      | UINT         | GetEnhMetaFileDescriptionW(_In_ HENHMETAFILE hemf, _In_ UINT cchBuffer, _Out_ LPWSTR lpszDescription)                                                    |
| tuitui      | UINT         | GetEnhMetaFileHeader(_In_ HENHMETAFILE hemf, _In_ UINT cbBuffer, _Out_ LPENHMETAHHEADER lpemh)                                                           |

|              |              |                                                                                                                                                                     |
|--------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitui       | UINT         | GetEnhMetaFilePaletteEntries(_In_ HENHMETAFILE hemf, _In_ UINT cEntries, _Out_ LPPALETTEENTRY lppe)                                                                 |
| tuitui       | UINT         | GetEnhMetaFilePixelFormat(HENHMETAFILE hemf, DWORD cbBuffer, const PIXELFORMATDESCRIPTOR *ppfd)                                                                     |
| wt           | HENHMETAFILE | GetEnhMetaFileW(_In_ LPCWSTR lpszMetaFile)                                                                                                                          |
| tuiuituiui   | DWORD        | GetFontData(_In_ HDC hdc, _In_ DWORD dwTable, _In_ DWORD dwOffset, _Out_ LPVOID lpvBuffer, _In_ DWORD cbData)                                                       |
| tui          | DWORD        | GetFontLanguageInfo(_In_ HDC hdc)                                                                                                                                   |
| ttui         | DWORD        | GetFontUnicodeRanges(_In_ HDC hdc, _Out_ LPGLYPHSET lpgs)                                                                                                           |
| tsituiui     | DWORD        | GetGlyphIndices(_In_ HDC hdc, _In_ LPCTSTR lpstr, _In_ int c, _Out_ LPWORD pgi, _In_ DWORD fl)                                                                      |
| taituiui     | DWORD        | GetGlyphIndicesA(_In_ HDC hdc, _In_ LPCSTR lpstr, _In_ int c, _Out_ LPWORD pgi, _In_ DWORD fl)                                                                      |
| twituiui     | DWORD        | GetGlyphIndicesW(_In_ HDC hdc, _In_ LPCWSTR lpstr, _In_ int c, _Out_ LPWORD pgi, _In_ DWORD fl)                                                                     |
| tuiuituittui | DWORD        | GetGlyphOutline(_In_ HDC hdc, _In_ UINT uChar, _In_ UINT uFormat, _Out_ LPGLYPHMETRICS lpgm, _In_ DWORD cbBuffer, _Out_ LPVOID lpvBuffer, _In_ const MAT2 *lpmat2)  |
| tuiuituittui | DWORD        | GetGlyphOutlineA(_In_ HDC hdc, _In_ UINT uChar, _In_ UINT uFormat, _Out_ LPGLYPHMETRICS lpgm, _In_ DWORD cbBuffer, _Out_ LPVOID lpvBuffer, _In_ const MAT2 *lpmat2) |
| tuiuituittui | DWORD        | GetGlyphOutlineW(_In_ HDC hdc, _In_ UINT uChar, _In_ UINT uFormat, _Out_ LPGLYPHMETRICS lpgm, _In_ DWORD cbBuffer, _Out_ LPVOID lpvBuffer, _In_ const MAT2 *lpmat2) |
| ti           | int          | GetGraphicsMode(_In_ HDC hdc)                                                                                                                                       |
| ttsi         | BOOL         | GetICMProfile(HDC hdc, LPDWORD lpcbName, LPTSTR lpszFilename)                                                                                                       |
| ttai         | BOOL         | GetICMProfileA(HDC hdc, LPDWORD lpcbName, LPSTR lpszFilename)                                                                                                       |
| ttwi         | BOOL         | GetICMProfileW(HDC hdc, LPDWORD lpcbName, LPWSTR lpszFilename)                                                                                                      |
| tuitui       | DWORD        | GetKerningPairs(_In_ HDC hdc, _In_ DWORD nNumPairs, _Out_ LPKERNINGPAIR lpkrnpair)                                                                                  |
| tuitui       | DWORD        | GetKerningPairsA(_In_ HDC hdc, _In_ DWORD nNumPairs, _Out_ LPKERNINGPAIR lpkrnpair)                                                                                 |
|              |              |                                                                                                                                                                     |

|          |          |                                                                                                             |
|----------|----------|-------------------------------------------------------------------------------------------------------------|
| tuitui   | DWORD    | GetKerningPairsW(_In_ HDC hdc, _In_ DWORD nNumPairs, _Out_ LPKERNINGPAIR lpkrnpair)                         |
| tui      | DWORD    | GetLayout(_In_ HDC hdc)                                                                                     |
| ttuii    | BOOL     | GetLogColorSpace(HCOLORSPACE hColorSpace, LPLOGCOLORSPACE lpBuffer, DWORD nSize)                            |
| ttuii    | BOOL     | GetLogColorSpaceA(HCOLORSPACE hColorSpace, LPLOGCOLORSPACE lpBuffer, DWORD nSize)                           |
| ttuii    | BOOL     | GetLogColorSpaceW(HCOLORSPACE hColorSpace, LPLOGCOLORSPACE lpBuffer, DWORD nSize)                           |
| ti       | int      | GetMapMode(_In_ HDC hdc)                                                                                    |
| tuitui   | UINT     | GetMetaFileBitsEx(_In_ HMETAFILE hmf, _In_ UINT nSize, _Out_ LPVOID lpvData)                                |
| tii      | int      | GetMetaRgn(_In_ HDC hdc, _In_ HRGN hrgn)                                                                    |
| tii      | BOOL     | GetMiterLimit(_In_ HDC hdc, _Out_ PFLOAT peLimit)                                                           |
| tuiui    | COLORREF | GetNearestColor(_In_ HDC hdc, _In_ COLORREF crColor)                                                        |
| tuiui    | UINT     | GetNearestPaletteIndex(_In_ HPALETTE hpal, _In_ COLORREF crColor)                                           |
| wti      | int      | GetNumberOfPhysicalMonitors(__in UNICODE_STRING *pstrDeviceName, __out LPDWORD pdwNumberOfPhysicalMonitors) |
| titi     | int      | GetObject(_In_ HGDIOBJ hgdiobj, _In_ int cbBuffer, _Out_ LPVOID lpvObject)                                  |
| titi     | int      | GetObjectA(_In_ HGDIOBJ hgdiobj, _In_ int cbBuffer, _Out_ LPVOID lpvObject)                                 |
| tui      | DWORD    | GetObjectType(_In_ HGDIOBJ h)                                                                               |
| titi     | int      | GetObjectW(_In_ HGDIOBJ hgdiobj, _In_ int cbBuffer, _Out_ LPVOID lpvObject)                                 |
| tuitui   | UINT     | GetOutlineTextMetrics(_In_ HDC hdc, _In_ UINT cbData, _Out_opt_ LPOUTLINETEXTMETRIC lpOTM)                  |
| tuitui   | UINT     | GetOutlineTextMetricsA(_In_ HDC hdc, _In_ UINT cbData, _Out_opt_ LPOUTLINETEXTMETRIC lpOTM)                 |
| tuitui   | UINT     | GetOutlineTextMetricsW(_In_ HDC hdc, _In_ UINT cbData, _Out_opt_ LPOUTLINETEXTMETRIC lpOTM)                 |
| tuiuitui | UINT     | GetPaletteEntries(_In_ HPALETTE hpal, _In_ UINT iStartIndex, _In_ UINT nEntries, _Out_ LPPALETTEENTRY lppe) |
| ttii     | int      | GetPath(_In_ HDC hdc, _Out_ LPPOINT lpPoints, _Out_ LPBYTE lpTypes, _In_ int nSize)                         |
| tuiwi    | int      | GetPhysicalMonitorDescriptor(__in HANDLE hMonitor, __in DWORD                                               |

|          |          |                                                                                                                                                                                        |
|----------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |          | dwPhysicalMonitorDescriptionSizeInChars, __out LPWSTR szPhysicalMonitorDescription)                                                                                                    |
| wuitti   | int      | GetPhysicalMonitor(__in UNICODE_STRING *pstrDeviceName, __in DWORD dwPhysicalMonitorArraySize, __out DWORD *pdwNumPhysicalMonitorHandlesInArray, __out HANDLE *phPhysicalMonitorArray) |
| tiiui    | COLORREF | GetPixel(_In_ HDC hdc, _In_ int nXPos, _In_ int nYPos)                                                                                                                                 |
| ti       | int      | GetPixelFormat(HDC hdc)                                                                                                                                                                |
| ti       | int      | GetPolyFillMode(_In_ HDC hdc)                                                                                                                                                          |
| ttii     | int      | GetRandomRgn(_In_ HDC hdc, _In_ HRGN hrgn, _In_ INT iNum)                                                                                                                              |
| tuii     | BOOL     | GetRasterizerCaps(_Out_ LPRASTERIZER_STATUS lprs, _In_ UINT cb)                                                                                                                        |
| tuitui   | DWORD    | GetRegionData(_In_ HRGN hRgn, _In_ DWORD dwCount, _Out_ LPRGNDATA lpRgnData)                                                                                                           |
| tii      | int      | GetRgnBox(_In_ HRGN hrgn, _Out_ LPRECT lprc)                                                                                                                                           |
| ti       | int      | GetROP2(_In_ HDC hdc)                                                                                                                                                                  |
| it       | HGDIOBJ  | GetStockObject(_In_ int fnObject)                                                                                                                                                      |
| ti       | int      | GetStretchBltMode(_In_ HDC hdc)                                                                                                                                                        |
| tuiuitui | UINT     | GetSystemPaletteEntries(_In_ HDC hdc, _In_ UINT iStartIndex, _In_ UINT nEntries, _Out_ LPPALETTEENTRY lppe)                                                                            |
| tui      | UINT     | GetSystemPaletteUse(_In_ HDC hdc)                                                                                                                                                      |
| tui      | UINT     | GetTextAlign(_In_ HDC hdc)                                                                                                                                                             |
| ti       | int      | GetTextCharacterExtra(_In_ HDC hdc)                                                                                                                                                    |
| ti       | int      | GetTextCharset(_In_ HDC hdc)                                                                                                                                                           |
| ttuii    | int      | GetTextCharsetInfo(_In_ HDC hdc, _Out_opt_ LPFONTSIGNATURE lpSig, _In_ DWORD dwFlags)                                                                                                  |
| tui      | COLORREF | GetTextColor(_In_ HDC hdc)                                                                                                                                                             |
| tsiitti  | BOOL     | GetTextExtentExPoint(_In_ HDC hdc, _In_ LPCTSTR lpszStr, _In_ int cchString, _In_ int nMaxExtent, _Out_ LPINT lpnFit, _Out_ LPINT alpDx, _Out_ LPSIZE lpSize)                          |
| taiitti  | BOOL     | GetTextExtentExPointA(_In_ HDC hdc, _In_ LPCSTR lpszStr, _In_ int cchString, _In_ int nMaxExtent, _Out_ LPINT lpnFit, _Out_ LPINT alpDx, _Out_ LPSIZE lpSize)                          |
| ttiiitti | BOOL     | GetTextExtentExPointI(_In_ HDC hdc, _In_ LPWORD pgiIn, _In_ int cgi, _In_ int nMaxExtent, _Out_ LPINT lpnFit, _Out_ LPINT alpDx, _Out_ LPSIZE lpSize)                                  |

|          |      |                                                                                                                                                                |
|----------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twiitti  | BOOL | GetTextExtentExPointW(_In_ HDC hdc, _In_ LPCWSTR lpszStr, _In_ int cchString, _In_ int nMaxExtent, _Out_ LPINT lpnFit, _Out_ LPINT alpDx, _Out_ LPSIZE lpSize) |
| tsiti    | BOOL | GetTextExtentPoint(_In_ HDC hdc, _In_ LPCTSTR lpString, _In_ int cbString, _Out_ LPSIZE lpSize)                                                                |
| tsiti    | BOOL | GetTextExtentPoint32(_In_ HDC hdc, _In_ LPCTSTR lpString, _In_ int c, _Out_ LPSIZE lpSize)                                                                     |
| taiti    | BOOL | GetTextExtentPoint32A(_In_ HDC hdc, _In_ LPCSTR lpString, _In_ int c, _Out_ LPSIZE lpSize)                                                                     |
| twiti    | BOOL | GetTextExtentPoint32W(_In_ HDC hdc, _In_ LPCWSTR lpString, _In_ int c, _Out_ LPSIZE lpSize)                                                                    |
| taiti    | BOOL | GetTextExtentPointA(_In_ HDC hdc, _In_ LPCSTR lpString, _In_ int cbString, _Out_ LPSIZE lpSize)                                                                |
| ttiti    | BOOL | GetTextExtentPointI(_In_ HDC hdc, _In_ LPWORD pgiIn, _In_ int cgi, _Out_ LPSIZE lpSize)                                                                        |
| twiti    | BOOL | GetTextExtentPointW(_In_ HDC hdc, _In_ LPCWSTR lpString, _In_ int cbString, _Out_ LPSIZE lpSize)                                                               |
| tisi     | int  | GetTextFace(_In_ HDC hdc, _In_ int nCount, _Out_ LPTSTR lpFaceName)                                                                                            |
| tiai     | int  | GetTextFaceA(_In_ HDC hdc, _In_ int nCount, _Out_ LPSTR lpFaceName)                                                                                            |
| tiwi     | int  | GetTextFaceW(_In_ HDC hdc, _In_ int nCount, _Out_ LPWSTR lpFaceName)                                                                                           |
| titi     | BOOL | GetTextMetrics(_In_ HDC hdc, _Out_ LPTEXTMETRIC lptm)                                                                                                          |
| titi     | BOOL | GetTextMetricsA(_In_ HDC hdc, _Out_ LPTEXTMETRIC lptm)                                                                                                         |
| titi     | BOOL | GetTextMetricsW(_In_ HDC hdc, _Out_ LPTEXTMETRIC lptm)                                                                                                         |
| titi     | BOOL | GetViewportExtEx(_In_ HDC hdc, _Out_ LPSIZE lpSize)                                                                                                            |
| titi     | BOOL | GetViewportOrgEx(_In_ HDC hdc, _Out_ LPPOINT lpPoint)                                                                                                          |
| titi     | BOOL | GetWindowExtEx(_In_ HDC hdc, _Out_ LPSIZE lpSize)                                                                                                              |
| titi     | BOOL | GetWindowOrgEx(_In_ HDC hdc, _Out_ LPPOINT lpPoint)                                                                                                            |
| tuititui | UINT | GetWinMetaFileBits(_In_ HENHMETAFILE hemf, _In_ UINT cbBuffer, _Out_ LPBYTE lpbBuffer, _In_ INT fnMapMode, _In_ HDC hdcRef)                                    |
| titi     | BOOL | GetWorldTransform(_In_ HDC hdc, _Out_ LPXFORM lpXform)                                                                                                         |
|          |      | IntersectClipRect(_In_ HDC hdc, _In_ int nLeftRect, _In_                                                                                                       |

|              |      |                                                                                                                                                                                                                                 |
|--------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiiii        | int  | int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                                                                                                                                                                        |
| tii          | BOOL | <i>InvertRgn</i> (_In_ HDC hdc, _In_ HRGN hrgn)                                                                                                                                                                                 |
| iiiiiti      | BOOL | <i>LineDDA</i> (_In_ int nXStart, _In_ int nYStart, _In_ int nXEnd, _In_ int nYEnd, _In_ LINEDDAPROC lpLineFunc, _In_ LPARAM lpData)                                                                                            |
| tiii         | BOOL | <i>LineTo</i> (_In_ HDC hdc, _In_ int nXEnd, _In_ int nYEnd)                                                                                                                                                                    |
| tii          | BOOL | <i>LPtoDP</i> (_In_ HDC hdc, _Inout_ LPPOINT lpPoints, _In_ int nCount)                                                                                                                                                         |
| tiiiiitiuiii | BOOL | <i>MaskBlt</i> (_In_ HDC hdcDest, _In_ int nXDest, _In_ int nYDest, _In_ int nWidth, _In_ int nHeight, _In_ HDC hdcSrc, _In_ int nXSrc, _In_ int nYSrc, _In_ HBITMAP hbmMask, _In_ int xMask, _In_ int yMask, _In_ DWORD dwRop) |
| ttuii        | BOOL | <i>ModifyWorldTransform</i> (_In_ HDC hdc, _In_ const XFORM *lpXform, _In_ DWORD iMode)                                                                                                                                         |
| tiiti        | BOOL | <i>MoveToEx</i> (_In_ HDC hdc, _In_ int X, _In_ int Y, _Out_ LPPOINT lpPoint)                                                                                                                                                   |
| tiii         | int  | <i>OffsetClipRgn</i> (_In_ HDC hdc, _In_ int nXOffset, _In_ int nYOffset)                                                                                                                                                       |
| tiii         | int  | <i>OffsetRgn</i> (_In_ HRGN hrgn, _In_ int nXOffset, _In_ int nYOffset)                                                                                                                                                         |
| tiiti        | BOOL | <i>OffsetViewportOrgEx</i> (_In_ HDC hdc, _In_ int nXOffset, _In_ int nYOffset, _Out_ LPPOINT lpPoint)                                                                                                                          |
| tiiti        | BOOL | <i>OffsetWindowOrgEx</i> (_In_ HDC hdc, _In_ int nXOffset, _In_ int nYOffset, _Out_ LPPOINT lpPoint)                                                                                                                            |
| tii          | BOOL | <i>PaintRgn</i> (_In_ HDC hdc, _In_ HRGN hrgn)                                                                                                                                                                                  |
| tiiiiuiii    | BOOL | <i>PatBlt</i> (_In_ HDC hdc, _In_ int nXLeft, _In_ int nYLeft, _In_ int nWidth, _In_ int nHeight, _In_ DWORD dwRop)                                                                                                             |
| tt           | HRGN | <i>PathToRegion</i> (_In_ HDC hdc)                                                                                                                                                                                              |
| tiiiiiiiiii  | BOOL | <i>Pie</i> (_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect, _In_ int nXRadial1, _In_ int nYRadial1, _In_ int nXRadial2, _In_ int nYRadial2)                                     |
| ttti         | BOOL | <i>PlayEnhMetaFile</i> (_In_ HDC hdc, _In_ HENHMETAFILE hemf, _In_ const RECT *lpRect)                                                                                                                                          |
| tttuii       | BOOL | <i>PlayEnhMetaFileRecord</i> (_In_ HDC hdc, _In_ LPHANDLETABLE lpHandletable, _In_ const ENHMETAFILE_RECORD *lpEnhMetaRecord, _In_ UINT nHandles)                                                                               |
| tii          | BOOL | <i>PlayMetaFile</i> (_In_ HDC hdc, _In_ HMETAFILE hmf)                                                                                                                                                                          |
| tttuii       | BOOL | <i>PlayMetaFileRecord</i> (_In_ HDC hdc, _In_ LPHANDLETABLE lpHandletable, _In_                                                                                                                                                 |

|           |      |                                                                                                                                                                                               |
|-----------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |      | LPMETARECORD lpMetaRecord, _In_ UINT nHandles)                                                                                                                                                |
| ttiiiiiii | BOOL | PlgBlt(_In_ HDC hdcDest, _In_ const POINT *lpPoint, _In_ HDC hdcSrc, _In_ int nXSrc, _In_ int nYSrc, _In_ int nWidth, _In_ int nHeight, _In_ HBITMAP hbmMask, _In_ int xMask, _In_ int yMask) |
| ttuii     | BOOL | PolyBezier(_In_ HDC hdc, _In_ const POINT *lppt, _In_ DWORD cPoints)                                                                                                                          |
| ttuii     | BOOL | PolyBezierTo(_In_ HDC hdc, _In_ const POINT *lppt, _In_ DWORD cCount)                                                                                                                         |
| ttii      | BOOL | PolyDraw(_In_ HDC hdc, _In_ const POINT *lppt, _In_ const BYTE *lpbTypes, _In_ int cCount)                                                                                                    |
| ttii      | BOOL | Polygon(_In_ HDC hdc, _In_ const POINT *lpPoints, _In_ int nCount)                                                                                                                            |
| ttii      | BOOL | Polyline(_In_ HDC hdc, _In_ const POINT *lppt, _In_ int cPoints)                                                                                                                              |
| ttuii     | BOOL | PolylineTo(_In_ HDC hdc, _In_ const POINT *lppt, _In_ DWORD cCount)                                                                                                                           |
| ttii      | BOOL | PolyPolygon(_In_ HDC hdc, _In_ const POINT *lpPoints, _In_ const INT *lpPolyCounts, _In_ int nCount)                                                                                          |
| ttuii     | BOOL | PolyPolyline(_In_ HDC hdc, _In_ const POINT *lppt, _In_ const DWORD *lpdwPolyPoints, _In_ DWORD cCount)                                                                                       |
| ttii      | BOOL | PolyTextOut(_In_ HDC hdc, _In_ const POLYTEXT *pptxt, _In_ int cStrings)                                                                                                                      |
| ttii      | BOOL | PolyTextOutA(_In_ HDC hdc, _In_ const POLYTEXT *pptxt, _In_ int cStrings)                                                                                                                     |
| ttii      | BOOL | PolyTextOutW(_In_ HDC hdc, _In_ const POLYTEXT *pptxt, _In_ int cStrings)                                                                                                                     |
| tiii      | BOOL | PtInRegion(_In_ HRGN hrgn, _In_ int X, _In_ int Y)                                                                                                                                            |
| tiii      | BOOL | PtVisible(_In_ HDC hdc, _In_ int X, _In_ int Y)                                                                                                                                               |
| tui       | UINT | RealizePalette(_In_ HDC hdc)                                                                                                                                                                  |
| tiiii     | BOOL | Rectangle(_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                                                                                     |
| tii       | BOOL | RectInRegion(_In_ HRGN hrgn, _In_ const RECT *lprc)                                                                                                                                           |
| tii       | BOOL | RectVisible(_In_ HDC hdc, _In_ const RECT *lprc)                                                                                                                                              |
| ti        | BOOL | RemoveFontMemResourceEx(_In_ HANDLE fh)                                                                                                                                                       |
| si        | BOOL | RemoveFontResource(_In_ LPCTSTR lpFileName)                                                                                                                                                   |
| ai        | BOOL | RemoveFontResourceA(_In_ LPCSTR lpFileName)                                                                                                                                                   |
| suiti     | BOOL | RemoveFontResourceEx(_In_ LPCTSTR lpFileName, _In_ DWORD fl, _In_ PVOID pdv)                                                                                                                  |

|          |          |                                                                                                                                              |
|----------|----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| auiti    | BOOL     | RemoveFontResourceExA(_In_ LPCSTR lpFileName, _In_ DWORD fl, _In_ PVOID pdv)                                                                 |
| wuiti    | BOOL     | RemoveFontResourceExW(_In_ LPCWSTR lpFileName, _In_ DWORD fl, _In_ PVOID pdv)                                                                |
| wi       | BOOL     | RemoveFontResourceW(_In_ LPCWSTR lpFileName)                                                                                                 |
| ttt      | HDC      | ResetDC(_In_ HDC hdc, _In_ const DEVMODE *lpInitData)                                                                                        |
| ttt      | HDC      | ResetDCA(_In_ HDC hdc, _In_ const DEVMODE *lpInitData)                                                                                       |
| ttt      | HDC      | ResetDCW(_In_ HDC hdc, _In_ const DEVMODE *lpInitData)                                                                                       |
| tuii     | BOOL     | ResizePalette(_In_ HPALETTE hpal, _In_ UINT nEntries)                                                                                        |
| tii      | BOOL     | RestoreDC(_In_ HDC hdc, _In_ int nSavedDC)                                                                                                   |
| tiiiiiii | BOOL     | RoundRect(_In_ HDC hdc, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect, _In_ int nWidth, _In_ int nHeight) |
| ti       | int      | SaveDC(_In_ HDC hdc)                                                                                                                         |
| tiiiiti  | BOOL     | ScaleViewportExtEx(_In_ HDC hdc, _In_ int Xnum, _In_ int Xdenom, _In_ int Ynum, _In_ int Ydenom, _Out_ LPSIZE lpSize)                        |
| tiiiiti  | BOOL     | ScaleWindowExtEx(_In_ HDC hdc, _In_ int Xnum, _In_ int Xdenom, _In_ int Ynum, _In_ int Ydenom, _Out_ LPSIZE lpSize)                          |
| tii      | BOOL     | SelectClipPath(_In_ HDC hdc, _In_ int iMode)                                                                                                 |
| tii      | int      | SelectClipRgn(_In_ HDC hdc, _In_ HRGN hrgn)                                                                                                  |
| ttt      | HGDIOBJ  | SelectObject(_In_ HDC hdc, _In_ HGDIOBJ hgdiobj)                                                                                             |
| ttit     | HPALETTE | SelectPalette(_In_ HDC hdc, _In_ HPALETTE hpal, _In_ BOOL bForceBackground)                                                                  |
| tii      | int      | SetAbortProc(_In_ HDC hdc, _In_ ABORTPROC lpAbortProc)                                                                                       |
| tii      | int      | SetArcDirection(_In_ HDC hdc, _In_ int ArcDirection)                                                                                         |
| tuitui   | LONG     | SetBitmapBits(_In_ HBITMAP hbitmap, _In_ DWORD cBytes, _In_ const VOID *lpBits)                                                              |
| tiiti    | BOOL     | SetBitmapDimensionEx(_In_ HBITMAP hbitmap, _In_ int nWidth, _In_ int nHeight, _Out_ LPSIZE lpSize)                                           |
| tuiui    | COLORREF | SetBkColor(_In_ HDC hdc, _In_ COLORREF crColor)                                                                                              |
| tii      | int      | SetBkMode(_In_ HDC hdc, _In_ int iBkMode)                                                                                                    |
| ttuiui   | UINT     | SetBoundsRect(_In_ HDC hdc, _In_ const RECT *lprcBounds, _In_ UINT flags)                                                                    |
|          |          | SetBrushOrgEx(_In_ HDC hdc, _In_ int nXOrg, _In_ int                                                                                         |

|                 |              |                                                                                                                                                                                                                                                                  |
|-----------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiiti           | BOOL         | nYOrg, _Out_ LPPOINT lppt)                                                                                                                                                                                                                                       |
| tii             | BOOL         | SetColorAdjustment(_In_ HDC hdc, _In_ const COLORADJUSTMENT *lpca)                                                                                                                                                                                               |
| tii             | HCOLORSPACE  | SetColorSpace(HDC hdc, HCOLORSPACE hColorSpace)                                                                                                                                                                                                                  |
| tuiui           | COLORREF     | SetDCBrushColor(_In_ HDC hdc, _In_ COLORREF crColor)                                                                                                                                                                                                             |
| tuiui           | COLORREF     | SetDCPenColor(_In_ HDC hdc, _In_ COLORREF crColor)                                                                                                                                                                                                               |
| tii             | BOOL         | SetDeviceGammaRamp(HDC hdc, LPVOID lpRamp)                                                                                                                                                                                                                       |
| tuiuitui        | UINT         | SetDIBColorTable(_In_ HDC hdc, _In_ UINT uStartIndex, _In_ UINT cEntries, _In_ const RGBQUAD *pColors)                                                                                                                                                           |
| ttuiiuttui      | int          | SetDIBits(_In_ HDC hdc, _In_ HBITMAP hbmp, _In_ UINT uStartScan, _In_ UINT cScanLines, _In_ const VOID *lpvBits, _In_ const BITMAPINFO *lpbmi, _In_ UINT fuColorUse)                                                                                             |
| tiuiuiiuiiuttui | int          | SetDIBitsToDevice(_In_ HDC hdc, _In_ int XDest, _In_ int YDest, _In_ DWORD dwWidth, _In_ DWORD dwHeight, _In_ int XSrc, _In_ int YSrc, _In_ UINT uStartScan, _In_ UINT cScanLines, _In_ const VOID *lpvBits, _In_ const BITMAPINFO *lpbmi, _In_ UINT fuColorUse) |
| uitt            | HENHMETAFILE | SetEnhMetaFileBits(_In_ UINT cbBuffer, _In_ const BYTE *lpData)                                                                                                                                                                                                  |
| tii             | int          | SetGraphicsMode(_In_ HDC hdc, _In_ int iMode)                                                                                                                                                                                                                    |
| tii             | int          | SetICMMode(HDC hdc, int iEnableICM)                                                                                                                                                                                                                              |
| tsi             | BOOL         | SetICMProfile(HDC hdc, LPTSTR lpFileName)                                                                                                                                                                                                                        |
| tai             | BOOL         | SetICMProfileA(HDC hdc, LPSTR lpFileName)                                                                                                                                                                                                                        |
| twi             | BOOL         | SetICMProfileW(HDC hdc, LPWSTR lpFileName)                                                                                                                                                                                                                       |
| tuiui           | DWORD        | SetLayout(_In_ HDC hdc, _In_ DWORD dwLayout)                                                                                                                                                                                                                     |
| tii             | int          | SetMapMode(_In_ HDC hdc, _In_ int fnMapMode)                                                                                                                                                                                                                     |
| tuiui           | DWORD        | SetMapperFlags(_In_ HDC hdc, _In_ DWORD dwFlag)                                                                                                                                                                                                                  |
| uitt            | HMETAFILE    | SetMetaFileBitsEx(_In_ UINT nSize, _In_ const BYTE *lpData)                                                                                                                                                                                                      |
| ti              | int          | SetMetaRgn(_In_ HDC hdc)                                                                                                                                                                                                                                         |
| tfti            | BOOL         | SetMiterLimit(_In_ HDC hdc, _In_ FLOAT eNewLimit, _Out_ PFLOAT peOldLimit)                                                                                                                                                                                       |
| tuiuitui        | UINT         | SetPaletteEntries(_In_ HPALETTE hpal, _In_ UINT iStart, _In_ UINT cEntries, _In_ const PALETTEENTRY *lppe)                                                                                                                                                       |

|              |              |                                                                                                                                                                                                                                               |
|--------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiuiui       | COLORREF     | SetPixel(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ COLORREF crColor)                                                                                                                                                                         |
| titi         | BOOL         | SetPixelFormat(HDC hdc, int iPixelFormat, const PIXELFORMATDESCRIPTOR *ppfd)                                                                                                                                                                  |
| tiuiii       | BOOL         | SetPixelV(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ COLORREF crColor)                                                                                                                                                                        |
| tii          | int          | SetPolyFillMode(_In_ HDC hdc, _In_ int iPolyFillMode)                                                                                                                                                                                         |
| tiiii        | BOOL         | SetRectRgn(_In_ HRGN hrgn, _In_ int nLeftRect, _In_ int nTopRect, _In_ int nRightRect, _In_ int nBottomRect)                                                                                                                                  |
| tii          | int          | SetROP2(_In_ HDC hdc, _In_ int fnDrawMode)                                                                                                                                                                                                    |
| tii          | int          | SetStretchBltMode(_In_ HDC hdc, _In_ int iStretchMode)                                                                                                                                                                                        |
| tuiui        | UINT         | SetSystemPaletteUse(_In_ HDC hdc, _In_ UINT uUsage)                                                                                                                                                                                           |
| tuiui        | UINT         | SetTextAlign(_In_ HDC hdc, _In_ UINT fMode)                                                                                                                                                                                                   |
| tii          | int          | SetTextCharacterExtra(_In_ HDC hdc, _In_ int nCharExtra)                                                                                                                                                                                      |
| tuiui        | COLORREF     | SetTextColor(_In_ HDC hdc, _In_ COLORREF crColor)                                                                                                                                                                                             |
| tiii         | BOOL         | SetTextJustification(_In_ HDC hdc, _In_ int nBreakExtra, _In_ int nBreakCount)                                                                                                                                                                |
| tiiti        | BOOL         | SetViewportExtEx(_In_ HDC hdc, _In_ int nXExtent, _In_ int nYExtent, _Out_ LPSIZE lpSize)                                                                                                                                                     |
| tiiti        | BOOL         | SetViewportOrgEx(_In_ HDC hdc, _In_ int X, _In_ int Y, _Out_ LPPOINT lpPoint)                                                                                                                                                                 |
| tiiti        | BOOL         | SetWindowExtEx(_In_ HDC hdc, _In_ int nXExtent, _In_ int nYExtent, _Out_ LPSIZE lpSize)                                                                                                                                                       |
| tiiti        | BOOL         | SetWindowOrgEx(_In_ HDC hdc, _In_ int X, _In_ int Y, _Out_ LPPOINT lpPoint)                                                                                                                                                                   |
| uitttt       | HENHMETAFILE | SetWinMetaFileBits(_In_ UINT cbBuffer, _In_ const BYTE *lpbBuffer, _In_ HDC hdcRef, _In_ const METAFILEPICT *lpmfp)                                                                                                                           |
| tii          | BOOL         | SetWorldTransform(_In_ HDC hdc, _In_ const XFORM *lpXform)                                                                                                                                                                                    |
| tii          | int          | StartDoc(_In_ HDC hdc, _In_ const DOCINFO *lpdi)                                                                                                                                                                                              |
| tii          | int          | StartDocA(_In_ HDC hdc, _In_ const DOCINFO *lpdi)                                                                                                                                                                                             |
| tii          | int          | StartDocW(_In_ HDC hdc, _In_ const DOCINFO *lpdi)                                                                                                                                                                                             |
| ti           | int          | StartPage(_In_ HDC hdc)                                                                                                                                                                                                                       |
| tiitiitiuiii | BOOL         | StretchBlt(_In_ HDC hdcDest, _In_ int nXOriginDest, _In_ int nYOriginDest, _In_ int nWidthDest, _In_ int nHeightDest, _In_ HDC hdcSrc, _In_ int nXOriginSrc, _In_ int nYOriginSrc, _In_ int nWidthSrc, _In_ int nHeightSrc, _In_ DWORD dwRop) |



# Gdiplus.dll

|            |          |                                                                                                               |
|------------|----------|---------------------------------------------------------------------------------------------------------------|
| tiiiiui    | GpStatus | GdiplAddPathArc (GpPath* path,REAL x,REAL y,REAL width,REAL height,REAL startAngle,REAL sweepAngle)           |
| tiiiiui    | GpStatus | GdiplAddPathArc1 (GpPath* path,INT x,INT y,INT width,INT height,REAL startAngle,REAL sweepAngle)              |
| tiiiiiiiui | GpStatus | GdiplAddPathBezier (GpPath* path,REAL x1,REAL y1,REAL x2,REAL y2,REAL x3,REAL y3,REAL x4,REAL y4)             |
| tiiiiiiiui | GpStatus | GdiplAddPathBezier1 (GpPath* path,INT x1,INT y1,INT x2,INT y2,INT x3,INT y3,INT x4,INT y4)                    |
| tiiui      | GpStatus | GdiplAddPathBeziern (GpPath* path,GDIPGpPointF* points,INT count)                                             |
| tiiui      | GpStatus | GdiplAddPathBeziern1 (GpPath* path,GDIPGpPoint* points,INT count)                                             |
| tiiui      | GpStatus | GdiplAddPathClosedCurve (GpPath* path,GDIPGpPointF* points,INT count)                                         |
| tiiui      | GpStatus | GdiplAddPathClosedCurve2 (GpPath* path,GDIPGpPointF* points,INT count,REAL tension)                           |
| tiiui      | GpStatus | GdiplAddPathClosedCurve21 (GpPath* path,GDIPGpPoint* points,INT count,REAL tension)                           |
| tiiui      | GpStatus | GdiplAddPathClosedCurve1 (GpPath* path,GDIPGpPoint* points,INT count)                                         |
| tiiui      | GpStatus | GdiplAddPathCurve (GpPath* path,GDIPGpPointF* points,INT count)                                               |
| tiiui      | GpStatus | GdiplAddPathCurve2 (GpPath* path,GDIPGpPointF* points,INT count,REAL tension)                                 |
| tiiui      | GpStatus | GdiplAddPathCurve21 (GpPath* path,GDIPGpPoint* points,INT count,REAL tension)                                 |
| tiiiiui    | GpStatus | GdiplAddPathCurve3 (GpPath* path,GDIPGpPointF* points,INT count,INT offset,INT numberOfSegments,REAL tension) |
| tiiiiui    | GpStatus | GdiplAddPathCurve31 (GpPath* path,GDIPGpPoint* points,INT count,INT offset,INT numberOfSegments,REAL tension) |
| tiiui      | GpStatus | GdiplAddPathCurve1 (GpPath* path,GDIPGpPoint* points,INT count)                                               |
| tiiiiui    | GpStatus | GdiplAddPathEllipse (GpPath* path,REAL x,REAL y,REAL width,REAL height)                                       |
| tiiiiui    | GpStatus | GdiplAddPathEllipse1 (GpPath* path,INT x,INT y,INT width,INT height)                                          |
| tiiiiui    | GpStatus | GdiplAddPathLine (GpPath* path,REAL x1,REAL y1,REAL x2,REAL y2)                                               |
| tiiui      | GpStatus | GdiplAddPathLine2 (GpPath* path,GDIPGpPointF* points,INT count)                                               |
| tiiui      | GpStatus | GdiplAddPathLine21 (GpPath* path,GDIPGpPoint* points,INT count)                                               |
| tiiiiui    | GpStatus | GdiplAddPathLine1 (GpPath* path,INT x1,INT y1,INT x2,INT y2)                                                  |
| tiiui      | GpStatus | GdiplAddPathPath (GpPath* path,GDIPGpPath* addingPath,BOOL connect)                                           |
| tiiiiiiiui | GpStatus | GdiplAddPathPie (GpPath* path,REAL x,REAL y,REAL width,REAL height,REAL startAngle,REAL sweepAngle)           |

|           |          |                                                                                                                                                              |
|-----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |          | height,REAL startAngle,REAL sweepAngle)                                                                                                                      |
| tiiiiui   | GpStatus | GdipAddPathPie(GpPath* path,INT x,INT y,INT width,INT height,REAL startAngle,REAL sweepAngle)                                                                |
| ttui      | GpStatus | GdipAddPathPolygon(GpPath* path,GDIPGpPointF* points,INT count)                                                                                              |
| ttui      | GpStatus | GdipAddPathPolygon(GpPath* path,GDIPGpPoint* points,INT count)                                                                                               |
| tiiiui    | GpStatus | GdipAddPathRectangle(GpPath* path,REAL x,REAL y,REAL width,REAL height)                                                                                      |
| tiiiui    | GpStatus | GdipAddPathRectangle(GpPath* path,INT x,INT y,INT width,INT height)                                                                                          |
| ttui      | GpStatus | GdipAddPathRectangles(GpPath* path,GDIPGpRectF* rects,INT count)                                                                                             |
| ttui      | GpStatus | GdipAddPathRectangles(GpPath* path,GDIPGpRect* rects,INT count)                                                                                              |
| ttitiitui | GpStatus | GdipAddPathString(GpPath* path,GDIPWCHAR* string,INT length,GDIPGpFontFamily* family,INT style,REAL emSize,GDIPRectF* layoutRect,GDIPGpStringFormat* format) |
| ttitiitui | GpStatus | GdipAddPathString(GpPath* path,GDIPWCHAR* string,INT length,GDIPGpFontFamily* family,INT style,REAL emSize,GDIPRect* layoutRect,GDIPGpStringFormat* format)  |
| tt        | void*    | GdipAlloc(size_t size)                                                                                                                                       |
| ttuitui   | GpStatus | GdipBeginContainer(GpGraphics* graphics,GDIPGpRectF* dstrect,GDIPGpRectF* srcrect,GpUnit unit,GraphicsContainer* state)                                      |
| ttui      | GpStatus | GdipBeginContainer2(GpGraphics* graphics,GraphicsContainer* state)                                                                                           |
| ttuitui   | GpStatus | GdipBeginContainer(GpGraphics* graphics,GDIPGpRect* dstrect,GDIPGpRect* srcrect,GpUnit unit,GraphicsContainer* state)                                        |
| tiiui     | GpStatus | GdipBitmapGetPixel(GpBitmap* bitmap,INT x,INT y,ARGB* color)                                                                                                 |
| ttuiitui  | GpStatus | GdipBitmapLockBits(GpBitmap* bitmap,GDIPGpRect* rect,UINT flags,PixelFormat format,BitmapData* lockedBitmapData)                                             |
| tiiui     | GpStatus | GdipBitmapSetPixel(GpBitmap* bitmap,INT x,INT y,ARGB color)                                                                                                  |
| tiiui     | GpStatus | GdipBitmapSetResolution(GpBitmap* bitmap,REAL xdpi,REAL ydpi)                                                                                                |
| ttui      | GpStatus | GdipBitmapUnlockBits(GpBitmap* bitmap,BitmapData* lockedBitmapData)                                                                                          |
| tui       | GpStatus | GdipClearPathMarkers(GpPath* path)                                                                                                                           |
| iiiiitui  | GpStatus | GdipCloneBitmapArea(REAL x,REAL y,REAL width,REAL height,PixelFormat format,GpBitmap* srcBitmap,GpBitmap* *dstBitmap)                                        |
| iiiiitui  | GpStatus | GdipCloneBitmapArea(INT x,INT y,INT width,INT height,PixelFormat format,GpBitmap* srcBitmap,GpBitmap* *dstBitmap)                                            |
| ttui      | GpStatus | GdipCloneBrush(GpBrush* brush,GpBrush* *cloneBrush)                                                                                                          |
| ttui      | GpStatus | GdipCloneCustomLineCap(GpCustomLineCap* customCap,GpCustomLineCap** clonedCap)                                                                               |
| ttui      | GpStatus | GdipCloneFont(GpFont* font,GpFont** cloneFont)                                                                                                               |
| ttui      | GpStatus | GdipCloneFontFamily(GpFontFamily* fontFamily,GpFontFamily* *clonedFontFamily)                                                                                |

|           |          |                                                                                                             |
|-----------|----------|-------------------------------------------------------------------------------------------------------------|
| ttui      | GpStatus | GdipCloneImage(GpImage* image,GpImage* *cloneImage)                                                         |
| ttui      | GpStatus | GdipCloneImageAttributes(GDIPGpImageAttributes* imageattr,GpImageAttributes* *cloneImageattr)               |
| ttui      | GpStatus | GdipCloneMatrix(GpMatrix* matrix,GpMatrix* *cloneMatrix)                                                    |
| ttui      | GpStatus | GdipClonePath(GpPath* path,GpPath* *clonePath)                                                              |
| ttui      | GpStatus | GdipClonePen(GpPen* pen,GpPen* *clonepen)                                                                   |
| ttui      | GpStatus | GdipCloneRegion(GpRegion* region,GpRegion* *cloneRegion)                                                    |
| ttui      | GpStatus | GdipCloneStringFormat(GDIPGpStringFormat* format,GpStringFormat* *newFormat)                                |
| tui       | GpStatus | GdipClosePathFigure(GpPath* path)                                                                           |
| tui       | GpStatus | GdipClosePathFiguref(GpPath* path)                                                                          |
| tttui     | GpStatus | GdipCombineRegionPath(GpRegion* region,GpPath* path,CombineMode combineMode)                                |
| tttui     | GpStatus | GdipCombineRegionRect(GpRegion* region,GDIPGpRectF* rect,CombineMode combineMode)                           |
| tttui     | GpStatus | GdipCombineRegionRectf(GpRegion* region,GDIPGpRect* rect,CombineMode combineMode)                           |
| tttui     | GpStatus | GdipCombineRegionRegion(GpRegion* region,GpRegion* region2,CombineMode combineMode)                         |
| tuitui    | GpStatus | GdipComment(GpGraphics* graphics,UINT sizeData,GDIPBYTE* data)                                              |
| iiitui    | GpStatus | GdipCreateAdjustableArrowCap(REAL height,REAL width,BOOL isFilled,GpAdjustableArrowCap* *cap)               |
| ttui      | GpStatus | GdipCreateBitmapFromDirectDrawSurface(IDirectDrawSurface7* surface,GpBitmap** bitmap)                       |
| ttui      | GpStatus | GdipCreateBitmapFromFile(GDIPWCHAR* filename,GpBitmap* *bitmap)                                             |
| ttui      | GpStatus | GdipCreateBitmapFromFileICM(GDIPWCHAR* filename,GpBitmap* *bitmap)                                          |
| tttui     | GpStatus | GdipCreateBitmapFromGdiDib(GDIPBITMAPINFO* gdiBitmapInfo,VOID* gdiBitmapData,GpBitmap** bitmap)             |
| iiittui   | GpStatus | GdipCreateBitmapFromGraphics(INT width,INT height,GpGraphics* target,GpBitmap** bitmap)                     |
| tttui     | GpStatus | GdipCreateBitmapFromHBITMAP(HBITMAP hbm,HPALETTE hpal,GpBitmap** bitmap)                                    |
| ttui      | GpStatus | GdipCreateBitmapFromHICON(HICON hicon,GpBitmap** bitmap)                                                    |
| tttui     | GpStatus | GdipCreateBitmapFromResource(HINSTANCE hInstance,GDIPWCHAR* lpBitmapName,GpBitmap** bitmap)                 |
| iiiiittui | GpStatus | GdipCreateBitmapFromScan0(INT width,INT height,INT stride,PixelFormat format,BYTE* scan0,GpBitmap** bitmap) |
| ttui      | GpStatus | GdipCreateBitmapFromStream(IStream* stream,GpBitmap* *bitmap)                                               |
|           |          |                                                                                                             |

|             |          |                                                                                                                                                                   |
|-------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttui        | GpStatus | GdipCreateBitmapFromStreamIcm(IStream* stream,GpBitmap* *bitmap)                                                                                                  |
| tttui       | GpStatus | GdipCreateCachedBitmap(GpBitmap* bitmap,GpGraphics* graphics,GpCachedBitmap* *cachedBitmap)                                                                       |
| ttuiitui    | GpStatus | GdipCreateCustomLineCap(GpPath* fillPath,GpPath* strokePath,GpLineCap* baseCap,REAL baseInset,GpCustomLineCap* *customCap)                                        |
| tiiuitui    | GpStatus | GdipCreateFont(GDIPGpFontFamily* fontFamily,REAL emSize,INT style,Unit unit,GpFont* *font)                                                                        |
| tttui       | GpStatus | GdipCreateFontFamilyFromName(GDIPWCHAR* name,GpFontCollection* fontCollection,GpFontFamily* *FontFamily)                                                          |
| ttui        | GpStatus | GdipCreateFontFromDc(HDC hdc,GpFont* *font)                                                                                                                       |
| tttui       | GpStatus | GdipCreateFontFromLogfont(HDC hdc,GDIPLOGFONTA* logfont,GpFont* *font)                                                                                            |
| tttui       | GpStatus | GdipCreateFontFromLogfontA(HDC hdc,GDIPLOGFONTA* logfont,GpFont* *font)                                                                                           |
| tttui       | GpStatus | GdipCreateFontFromLogfontW(HDC hdc,GDIPLOGFONTW* logfont,GpFont* *font)                                                                                           |
| ttui        | GpStatus | GdipCreateFromHdc(HDC hdc,GpGraphics* *graphics)                                                                                                                  |
| tttui       | GpStatus | GdipCreateFromHdc2(HDC hdc,HANDLE hDevice,GpGraphics* *graphics)                                                                                                  |
| ttui        | GpStatus | GdipCreateFromHwnd(HWND hwnd,GpGraphics* *graphics)                                                                                                               |
| ttui        | GpStatus | GdipCreateFromHwndIcm(HWND hwnd,GpGraphics* *graphics)                                                                                                            |
| t           | HPALETTE | GdipCreateHalftonePalette()                                                                                                                                       |
| uiuiuitui   | GpStatus | GdipCreateHatchBrush(GpHatchStyle hatchstyle,ARGB forecol,ARGB backcol,GpHatch* *brush)                                                                           |
| ttuiui      | GpStatus | GdipCreateHBITMAPFromBitmap(GpBitmap* bitmap,HBITMAP* hbmReturn,ARGB background)                                                                                  |
| ttui        | GpStatus | GdipCreateHICONFromBitmap(GpBitmap* bitmap,HICON* hbmReturn)                                                                                                      |
| tui         | GpStatus | GdipCreateImageAttributes(GpImageAttributes* *imageattr)                                                                                                          |
| ttuiuiuitui | GpStatus | GdipCreateLineBrush(GDIPGpPointF* point1,GDIPGpPointF* point2,ARGB color1,ARGB color2,GpWrapMode wrapMode,GpLineGradient* *lineGradient)                          |
| tuiuiiuitui | GpStatus | GdipCreateLineBrushFromRect(GDIPGpRectF* rect,ARGB color1,ARGB color2,LinearGradientMode mode,GpWrapMode wrapMode,GpLineGradient* *lineGradient)                  |
| tuiuiiuitui | GpStatus | GdipCreateLineBrushFromRectI(GDIPGpRect* rect,ARGB color1,ARGB color2,LinearGradientMode mode,GpWrapMode wrapMode,GpLineGradient* *lineGradient)                  |
| tuiuiiuitui | GpStatus | GdipCreateLineBrushFromRectWithAngle(GDIPGpRectF* rect,ARGB color1,ARGB color2,REAL angle,BOOL isAngleScalable,GpWrapMode wrapMode,GpLineGradient* *lineGradient) |
| tuiuiiuitui | GpStatus | GdipCreateLineBrushFromRectWithAngleI(GDIPGpRect* rect,ARGB color1,ARGB color2,REAL angle,BOOL isAngleScalable,GpWrapMode wrapMode,GpLineGradient* *lineGradient) |

|             |          |                                                                                                                                        |
|-------------|----------|----------------------------------------------------------------------------------------------------------------------------------------|
|             |          | wrapMode,GpLineGradient* *lineGradient)                                                                                                |
| ttuiuiuitui | GpStatus | GdipCreateLineBrush1(GDIPGpPoint* point1,GDIPGpPoint* point2,ARG color1,ARGB color2,GpWrapMode wrapMode,GpLineGradient* *lineGradient) |
| tui         | GpStatus | GdipCreateMatrix(GpMatrix* *matrix)                                                                                                    |
| iiiiitui    | GpStatus | GdipCreateMatrix2(REAL m11,REAL m12,REAL m21,REAL m22,REAL dx,REAL dy,GpMatrix* *matrix)                                               |
| ttui        | GpStatus | GdipCreateMatrix3(GDIPGpRectF* rect,GDIPGpPointF* dstplg,GpMatrix* *matrix)                                                            |
| ttui        | GpStatus | GdipCreateMatrix3I(GDIPGpRect* rect,GDIPGpPointF* dstplg,GpMatrix* *matrix)                                                            |
| titui       | GpStatus | GdipCreateMetafileFromEmf(HENHMETAFILE hEmf,BOOL deleteEmf,GpMetafile* *metafile)                                                      |
| ttui        | GpStatus | GdipCreateMetafileFromFile(GDIPWCHAR* file,GpMetafile* *metafile)                                                                      |
| ttui        | GpStatus | GdipCreateMetafileFromStream(IStream* stream,GpMetafile* *metafile)                                                                    |
| tittui      | GpStatus | GdipCreateMetafileFromWmf(HMETAFILE hWmf,BOOL deleteWmf,GDIPWmfPlaceableFileHeader* wmfPlaceableFileHeader,GpMetafile* *metafile)      |
| ttui        | GpStatus | GdipCreateMetafileFromWmfFile(GDIPWCHAR* file,GDIPWmfPlaceableFileHeader* wmfPlaceableFileHeader,GpMetafile* *metafile)                |
| uitui       | GpStatus | GdipCreatePath(GpFillMode brushMode,GpPath* *path)                                                                                     |
| ttuiuitui   | GpStatus | GdipCreatePath2(GDIPGpPointF*,GDIPBYTE*,INT,GpFillMode,GpPath* *path)                                                                  |
| ttuiuitui   | GpStatus | GdipCreatePath2I(GDIPGpPoint*,GDIPBYTE*,INT,GpFillMode,GpPath* *path)                                                                  |
| tiuitui     | GpStatus | GdipCreatePathGradient(GDIPGpPointF* points,INT count,GpWrapMode wrapMode,GpPathGradient* *polyGradient)                               |
| ttui        | GpStatus | GdipCreatePathGradientFromPath(GDIPGpPath* path,GpPathGradient* *polyGradient)                                                         |
| tiuitui     | GpStatus | GdipCreatePathGradientI(GDIPGpPoint* points,INT count,GpWrapMode wrapMode,GpPathGradient* *polyGradient)                               |
| ttui        | GpStatus | GdipCreatePathIter(GpPathIterator* *iterator,GpPath* path)                                                                             |
| uiiuitui    | GpStatus | GdipCreatePen1(ARGB color,REAL width,GpUnit unit,GpPen* *pen)                                                                          |
| tiuitui     | GpStatus | GdipCreatePen2(GpBrush* brush,REAL width,GpUnit unit,GpPen* *pen)                                                                      |
| tui         | GpStatus | GdipCreateRegion(GpRegion* *region)                                                                                                    |
| ttui        | GpStatus | GdipCreateRegionHrgn(HRGN hRgn,GpRegion* *region)                                                                                      |
| ttui        | GpStatus | GdipCreateRegionPath(GpPath* path,GpRegion* *region)                                                                                   |
| ttui        | GpStatus | GdipCreateRegionRect(GDIPGpRectF* rect,GpRegion* *region)                                                                              |
| ttui        | GpStatus | GdipCreateRegionRectI(GDIPGpRect* rect,GpRegion* *region)                                                                              |

|           |          |                                                                                                                                     |
|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| titui     | GpStatus | GdipCreateRegionRgnData(GDIPBYTE* regionData,INT size,GpRegion* *region)                                                            |
| uitui     | GpStatus | GdipCreateSolidFill(ARGB color,GpSolidFill* *brush)                                                                                 |
| tuitui    | GpStatus | GdipCreateStreamOnFile(GDIPWCHAR* filename,UINT access,IStream* *stream)                                                            |
| iuhtui    | GpStatus | GdipCreateStringFormat(INT formatAttributes,LANGID language,GpStringFormat* *format)                                                |
| tuitui    | GpStatus | GdipCreateTexture(GpImage* image,GpWrapMode wrapmode,GpTexture* *texture)                                                           |
| tuiiiitui | GpStatus | GdipCreateTexture2(GpImage* image,GpWrapMode wrapmode,REAL x,REAL y,REAL width,REAL height,GpTexture* *texture)                     |
| tuiiiitui | GpStatus | GdipCreateTexture2i(GpImage* image,GpWrapMode wrapmode,INT x,INT y,INT width,INT height,GpTexture* *texture)                        |
| tiiiitui  | GpStatus | GdipCreateTextureIA(GpImage* image,GDIPGpImageAttributes* imageAttributes,REAL x,REAL y,REAL width,REAL height,GpTexture* *texture) |
| tiiiitui  | GpStatus | GdipCreateTextureIAi(GpImage* image,GDIPGpImageAttributes* imageAttributes,INT x,INT y,INT width,INT height,GpTexture* *texture)    |
| tui       | GpStatus | GdipDeleteBrush(GpBrush* brush)                                                                                                     |
| tui       | GpStatus | GdipDeleteCachedBitmap(GpCachedBitmap* cachedBitmap)                                                                                |
| tui       | GpStatus | GdipDeleteCustomLineCap(GpCustomLineCap* customCap)                                                                                 |
| tui       | GpStatus | GdipDeleteFont(GpFont* font)                                                                                                        |
| tui       | GpStatus | GdipDeleteFontFamily(GpFontFamily* FontFamily)                                                                                      |
| tui       | GpStatus | GdipDeleteGraphics(GpGraphics* graphics)                                                                                            |
| tui       | GpStatus | GdipDeleteMatrix(GpMatrix* matrix)                                                                                                  |
| tui       | GpStatus | GdipDeletePath(GpPath* path)                                                                                                        |
| tui       | GpStatus | GdipDeletePathIter(GpPathIterator* iterator)                                                                                        |
| tui       | GpStatus | GdipDeletePen(GpPen* pen)                                                                                                           |
| tui       | GpStatus | GdipDeletePrivateFontCollection(GpFontCollection** fontCollection)                                                                  |
| tui       | GpStatus | GdipDeleteRegion(GpRegion* region)                                                                                                  |
| tui       | GpStatus | GdipDeleteStringFormat(GpStringFormat* format)                                                                                      |
| tui       | GpStatus | GdipDisposeImage(GpImage* image)                                                                                                    |
| tui       | GpStatus | GdipDisposeImageAttributes(GpImageAttributes* imageattr)                                                                            |
| tiiiiitui | GpStatus | GdipDrawArc(GpGraphics* graphics,GpPen* pen,REAL x,REAL y,REAL width,REAL height,REAL startAngle,REAL sweepAngle)                   |
| tiiiiitui | GpStatus | GdipDrawArcI(GpGraphics* graphics,GpPen* pen,INT x,INT y,INT width,REAL height,REAL startAngle,REAL sweepAngle)                     |
| tiiiiitui | GpStatus | GdipDrawBezier(GpGraphics* graphics,GpPen* pen,REAL x1,REAL                                                                         |

|             |          |                                                                                                                                                                 |
|-------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |          | y1,REAL x2,REAL y2,REAL x3,REAL y3,REAL x4,REAL y4)                                                                                                             |
| ttiiiiiiiui | GpStatus | GdipDrawBezier1(GpGraphics* graphics,GpPen* pen,INT x1,INT y1,INT x2,INT y2,INT x3,INT y3,INT x4,INT y4)                                                        |
| tttiui      | GpStatus | GdipDrawBeziers(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                                 |
| tttiui      | GpStatus | GdipDrawBeziers1(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                                |
| tttiui      | GpStatus | GdipDrawCachedBitmap(GpGraphics* graphics,GpCachedBitmap* cachedBitmap,INT x,INT y)                                                                             |
| tttiui      | GpStatus | GdipDrawClosedCurve(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                             |
| tttiui      | GpStatus | GdipDrawClosedCurve2(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,REAL tension)                                                               |
| tttiui      | GpStatus | GdipDrawClosedCurve21(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,REAL tension)                                                              |
| tttiui      | GpStatus | GdipDrawClosedCurve21(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,REAL tension)                                                              |
| tttiui      | GpStatus | GdipDrawClosedCurve21(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                           |
| tttiui      | GpStatus | GdipDrawCurve(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                                   |
| tttiui      | GpStatus | GdipDrawCurve2(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,REAL tension)                                                                     |
| tttiui      | GpStatus | GdipDrawCurve2(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,REAL tension)                                                                     |
| ttiiiiiiiui | GpStatus | GdipDrawCurve3(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,INT offset,INT numberOfSegments,REAL tension)                                     |
| ttiiiiiiiui | GpStatus | GdipDrawCurve31(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count,INT offset,INT numberOfSegments,REAL tension)                                    |
| tttiui      | GpStatus | GdipDrawCurve1(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                                  |
| tttttitui   | GpStatus | GdipDrawDriverString(GpGraphics* graphics,GDIPUINT16* text,INT length,GDIPGpFont* font,GDIPGpBrush* brush,GDIPPointF* positions,INT flags,GDIPGpMatrix* matrix) |
| ttiiiiiiiui | GpStatus | GdipDrawEllipse(GpGraphics* graphics,GpPen* pen,REAL x,REAL y,REAL width,REAL height)                                                                           |
| ttiiiiiiiui | GpStatus | GdipDrawEllipse(GpGraphics* graphics,GpPen* pen,INT x,INT y,INT width,INT height)                                                                               |
| ttiiui      | GpStatus | GdipDrawImage(GpGraphics* graphics,GpImage* image,REAL x,REAL y,REAL width,REAL height)                                                                         |
| ttiiui      | GpStatus | GdipDrawImage1(GpGraphics* graphics,GpImage* image,INT x,INT y)                                                                                                 |
| ttiiiiiiiui | GpStatus | GdipDrawImagePointRect(GpGraphics* graphics,GpImage* image,REAL x,REAL y,REAL srcx,REAL srcy,REAL srcwidth,REAL srcheight,GpUnit srcUnit)                       |
|             |          |                                                                                                                                                                 |

|                   |          |                                                                                                                                                                                                                                                               |
|-------------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttiiiiiiiui       | GpStatus | GdipDrawImagePointRectI(GpGraphics* graphics,GpImage* image,INT x,y,INT srcx,INT srcy,INT srcwidth,INT srcheight,GpUnit srcUnit)                                                                                                                              |
| tttiii            | GpStatus | GdipDrawImagePoints(GpGraphics* graphics,GpImage* image,GDIPGpPointF* dstpoints,INT count)                                                                                                                                                                    |
| tttiii            | GpStatus | GdipDrawImagePointsI(GpGraphics* graphics,GpImage* image,GDIPGpPoint* dstpoints,INT count)                                                                                                                                                                    |
| ttiiiiiiiuituitui | GpStatus | GdipDrawImagePointsRect(GpGraphics* graphics,GpImage* image,GDIPGpPointF* points,INT count,REAL srcx,REAL srcy,REAL srcwidth,REAL srcheight,GpUnit srcUnit,GDIPGpImageAttributes* imageAttributes,DrawImageAbort callback,VOID* callbackData)                 |
| ttiiiiiiiuituitui | GpStatus | GdipDrawImagePointsRectI(GpGraphics* graphics,GpImage* image,GDIPGpPoint* points,INT count,INT srcx,INT srcy,INT srcwidth,INT srcheight,GpUnit srcUnit,GDIPGpImageAttributes* imageAttributes,DrawImageAbort callback,VOID* callbackData)                     |
| ttiiiiui          | GpStatus | GdipDrawImageRect(GpGraphics* graphics,GpImage* image,REAL x,REAL y,REAL width,REAL height)                                                                                                                                                                   |
| ttiiiiui          | GpStatus | GdipDrawImageRectI(GpGraphics* graphics,GpImage* image,INT x,INT y,INT width,INT height)                                                                                                                                                                      |
| ttiiiiiiiuituitui | GpStatus | GdipDrawImageRectRect(GpGraphics* graphics,GpImage* image,REAL dstx,REAL dsty,REAL dstwidth,REAL dstheight,REAL srcx,REAL srcy,REAL srcwidth,REAL srcheight,GpUnit srcUnit,GDIPGpImageAttributes* imageAttributes,DrawImageAbort callback,VOID* callbackData) |
| ttiiiiiiiuituitui | GpStatus | GdipDrawImageRectRectI(GpGraphics* graphics,GpImage* image,INT dstx,INT dsty,INT dstwidth,INT dstheight,INT srcx,INT srcy,INT srcwidth,INT srcheight,GpUnit srcUnit,GDIPGpImageAttributes* imageAttributes,DrawImageAbort callback,VOID* callbackData)        |
| ttiiiiui          | GpStatus | GdipDrawLine(GpGraphics* graphics,GpPen* pen,REAL x1,REAL y1,REAL x2,REAL y2)                                                                                                                                                                                 |
| ttiiiiui          | GpStatus | GdipDrawLineI(GpGraphics* graphics,GpPen* pen,INT x1,INT y1,INT x2,INT y2)                                                                                                                                                                                    |
| tttiii            | GpStatus | GdipDrawLines(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                                                                                                                                 |
| tttiii            | GpStatus | GdipDrawLinesI(GpGraphics* graphics,GpPen* pen,GDIPGpPoint* points,INT count)                                                                                                                                                                                 |
| tttiii            | GpStatus | GdipDrawPath(GpGraphics* graphics,GpPen* pen,GpPath* path)                                                                                                                                                                                                    |
| ttiiiiiiiui       | GpStatus | GdipDrawPie(GpGraphics* graphics,GpPen* pen,REAL x,REAL y,REAL width,REAL height,REAL startAngle,REAL sweepAngle)                                                                                                                                             |
| ttiiiiiiiui       | GpStatus | GdipDrawPieI(GpGraphics* graphics,GpPen* pen,INT x,INT y,INT width,INT height,REAL startAngle,REAL sweepAngle)                                                                                                                                                |
| tttiii            | GpStatus | GdipDrawPolygon(GpGraphics* graphics,GpPen* pen,GDIPGpPointF* points,INT count)                                                                                                                                                                               |
| tttiii            | GpStatus | GdipDrawPolygonI(GpGraphics* graphics,GpPen* pen,GDIPGpPoint* points,INT count)                                                                                                                                                                               |

|           |          |                                                                                                                                                                                                                                                  |
|-----------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiiiiui   | GpStatus | GdipDrawRectangle(GpGraphics* graphics,GpPen* pen,REAL x,REAL y,REAL width,REAL height)                                                                                                                                                          |
| tiiiiui   | GpStatus | GdipDrawRectangleI(GpGraphics* graphics,GpPen* pen,INT x,INT y,INT width,INT height)                                                                                                                                                             |
| titiui    | GpStatus | GdipDrawRectangles(GpGraphics* graphics,GpPen* pen,GDIPGpRectF* rects,INT count)                                                                                                                                                                 |
| titiui    | GpStatus | GdipDrawRectanglesI(GpGraphics* graphics,GpPen* pen,GDIPGpRect* rects,INT count)                                                                                                                                                                 |
| ttitttui  | GpStatus | GdipDrawString(GpGraphics* graphics,GDIPWCHAR* string,INT length,GDIPGpFont* font,GDIPRectF* layoutRect,GDIPGpStringFormat* stringFormat,GDIPGpBrush* brush)                                                                                     |
| tuitiui   | UINT     | GdipEmfToWmfBis(HENHMETAFILE hemf,UINT cbData16,LPBYTE pData16,INT iMapMode,INT eFlags)                                                                                                                                                          |
| tuiui     | GpStatus | GdipEndContainer(GpGraphics* graphics,GraphicsContainer state)                                                                                                                                                                                   |
| tttttui   | GpStatus | GdipEnumerateMetafileDestPoint(GpGraphics* graphics,GDIPGpMetafile metafile,GDIPPointF & destPoint,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)                                                     |
| tttttui   | GpStatus | GdipEnumerateMetafileDestPointI(GpGraphics* graphics,GDIPGpMetafile metafile,GDIPPoint & destPoint,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)                                                     |
| ttitttui  | GpStatus | GdipEnumerateMetafileDestPoints(GpGraphics* graphics,GDIPGpMetafile metafile,GDIPPointF* destPoints,INT count,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)                                          |
| ttitttui  | GpStatus | GdipEnumerateMetafileDestPointsI(GpGraphics* graphics,GDIPGpMetafile metafile,GDIPPoint* destPoints,INT count,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)                                          |
| tttttui   | GpStatus | GdipEnumerateMetafileDestRect(GpGraphics* graphics,GDIPGpMetafile metafile,GDIPRectF & destRect,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)                                                        |
| tttttui   | GpStatus | GdipEnumerateMetafileDestRectI(GpGraphics* graphics,GDIPGpMetafile metafile,GDIPRect & destRect,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)                                                        |
| tttuittui | GpStatus | GdipEnumerateMetafileSrcRectDestPoint(GpGraphics* graphics,GDIPGpMetafile* metafile,GDIPPointF & destPoint,GDIPRectF srcRect,Unit srcUnit,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)              |
| tttuittui | GpStatus | GdipEnumerateMetafileSrcRectDestPointI(GpGraphics* graphics,GDIPGpMetafile* metafile,GDIPPoint & destPoint,GDIPRect & srcRect,Unit srcUnit,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)             |
| tttuittui | GpStatus | GdipEnumerateMetafileSrcRectDestPoints(GpGraphics* graphics,GDIPGpMetafile* metafile,GDIPPointF* destPoints,INT count,GDIPRectF & srcRect,Unit srcUnit,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes) |

|            |          |                                                                                                                                                                                                                                                                 |
|------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tttuitttui | GpStatus | <a href="#">GdipEnumerateMetafileSrcRectDestPoints</a> (GpGraphics* graphics,GDIPGpMetafile* metafile,GDIPPoint* destPoints,INT count,GDIPRect & srcRect,Unit srcUnit,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes) |
| tttuitttui | GpStatus | <a href="#">GdipEnumerateMetafileSrcRectDestRect</a> (GpGraphics* graphics,GDIPGpMetafile* metafile,GDIPRectF & destRect,GDIPRectF & srcRect,Unit srcUnit,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)             |
| tttuitttui | GpStatus | <a href="#">GdipEnumerateMetafileSrcRectDestRect</a> (GpGraphics* graphics,GDIPGpMetafile* metafile,GDIPRect & destRect,GDIPRect & srcRect,Unit srcUnit,EnumerateMetafileProc callback,VOID* callbackData,GDIPGpImageAttributes* imageAttributes)               |
| tttiii     | GpStatus | <a href="#">GdipFillClosedCurve</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPoint points,INT count)                                                                                                                                                          |
| tttiiiui   | GpStatus | <a href="#">GdipFillClosedCurve2</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPointF* points,INT count,REAL tension,GpFillMode fillMode)                                                                                                                      |
| tttiiiui   | GpStatus | <a href="#">GdipFillClosedCurve2I</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPoint points,INT count,REAL tension,GpFillMode fillMode)                                                                                                                       |
| tttiii     | GpStatus | <a href="#">GdipFillClosedCurveI</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPoint points,INT count)                                                                                                                                                         |
| tttiiiui   | GpStatus | <a href="#">GdipFillEllipse</a> (GpGraphics* graphics,GpBrush* brush,REAL x,REAL y,REAL width,REAL height)                                                                                                                                                      |
| tttiiiui   | GpStatus | <a href="#">GdipFillEllipseI</a> (GpGraphics* graphics,GpBrush* brush,INT x,INT y,INT width,INT height)                                                                                                                                                         |
| tttiii     | GpStatus | <a href="#">GdipFillPath</a> (GpGraphics* graphics,GpBrush* brush,GpPath* path)                                                                                                                                                                                 |
| tttiiiui   | GpStatus | <a href="#">GdipFillPie</a> (GpGraphics* graphics,GpBrush* brush,REAL x,REAL y,REAL width,REAL height,REAL startAngle,REAL sweepAngle)                                                                                                                          |
| tttiiiui   | GpStatus | <a href="#">GdipFillPieI</a> (GpGraphics* graphics,GpBrush* brush,INT x,INT y,INT width,INT height,REAL startAngle,REAL sweepAngle)                                                                                                                             |
| tttiiiui   | GpStatus | <a href="#">GdipFillPolygon</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPointF* points,INT count,GpFillMode fillMode)                                                                                                                                        |
| tttiii     | GpStatus | <a href="#">GdipFillPolygon2</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPointF* points,INT count)                                                                                                                                                           |
| tttiii     | GpStatus | <a href="#">GdipFillPolygon2I</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPoint* points,INT count)                                                                                                                                                           |
| tttiiiui   | GpStatus | <a href="#">GdipFillPolygonI</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpPoint* points,INT count,GpFillMode fillMode)                                                                                                                                        |
| tttiiiui   | GpStatus | <a href="#">GdipFillRectangle</a> (GpGraphics* graphics,GpBrush* brush,REAL x,REAL y,REAL width,REAL height)                                                                                                                                                    |
| tttiiiui   | GpStatus | <a href="#">GdipFillRectangleI</a> (GpGraphics* graphics,GpBrush* brush,INT x,INT y,INT width,INT height)                                                                                                                                                       |
| tttiii     | GpStatus | <a href="#">GdipFillRectangles</a> (GpGraphics* graphics,GpBrush* brush,GDIPGpRectF rects,INT count)                                                                                                                                                            |

|            |          |                                                                                                        |
|------------|----------|--------------------------------------------------------------------------------------------------------|
| tttuiu     | GpStatus | GdipFillRectangles(GpGraphics* graphics,GpBrush* brush,GDIPGpRects,INT count)                          |
| tttuiu     | GpStatus | GdipFillRegion(GpGraphics* graphics,GpBrush* brush,GpRegion* region)                                   |
| ttuiu      | GpStatus | GdipFlattenPath(GpPath* path,GpMatrix* matrix,REAL flatness)                                           |
| tuiu       | GpStatus | GdipFlush(GpGraphics* graphics,GpFlushIntention intention)                                             |
| ti         | void     | GdipFree(void* ptr)                                                                                    |
| ttuiu      | GpStatus | GdipGetAdjustableArrowCapFillState(GpAdjustableArrowCap* cap,BOOL fillState)                           |
| ttuiu      | GpStatus | GdipGetAdjustableArrowCapHeight(GpAdjustableArrowCap* cap,REAL height)                                 |
| ttuiu      | GpStatus | GdipGetAdjustableArrowCapMiddleInset(GpAdjustableArrowCap* cap,REAL* middleInset)                      |
| ttuiu      | GpStatus | GdipGetAdjustableArrowCapWidth(GpAdjustableArrowCap* cap,REAL* width)                                  |
| tuiuittuiu | GpStatus | GdipGetAllPropertyItems(GpImage* image,UINT totalBufferSize,UINT numProperties,PropertyItem* allItems) |
| ttuiu      | GpStatus | GdipGetBrushType(GpBrush* brush,GpBrushType* type)                                                     |
| ttuiu      | GpStatus | GdipGetCellAscent(GDIPGpFontFamily* family,INT style,UINT16* CellAscent)                               |
| ttuiu      | GpStatus | GdipGetCellDescent(GDIPGpFontFamily* family,INT style,UINT16* CellDescent)                             |
| ttuiu      | GpStatus | GdipGetClip(GpGraphics* graphics,GpRegion* region)                                                     |
| ttuiu      | GpStatus | GdipGetClipBounds(GpGraphics* graphics,GpRectF* rect)                                                  |
| ttuiu      | GpStatus | GdipGetClipBoundsI(GpGraphics* graphics,GpRect* rect)                                                  |
| ttuiu      | GpStatus | GdipGetCompositingMode(GpGraphics* graphics,CompositingMode* compositingMode)                          |
| ttuiu      | GpStatus | GdipGetCompositingQuality(GpGraphics* graphics,CompositingQuality* compositingQuality)                 |
| ttuiu      | GpStatus | GdipGetCustomLineCapBaseCap(GpCustomLineCap* customCap,GpLineCap* baseCap)                             |
| ttuiu      | GpStatus | GdipGetCustomLineCapBaseInset(GpCustomLineCap* customCap,REAL inset)                                   |
| tttuiu     | GpStatus | GdipGetCustomLineCapStrokeCaps(GpCustomLineCap* customCap,GpLineCap* startCap,GpLineCap* endCap)       |
| ttuiu      | GpStatus | GdipGetCustomLineCapStrokeJoin(GpCustomLineCap* customCap,GpLineJoin* lineJoin)                        |
| ttuiu      | GpStatus | GdipGetCustomLineCapType(GpCustomLineCap* customCap,CustomLineCapType* capType)                        |
| ttuiu      | GpStatus | GdipGetCustomLineCapWidthScale(GpCustomLineCap* customCap,REAL widthScale)                             |

|         |          |                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| ttui    | GpStatus | GdipGetDC(GpGraphics* graphics,HDC* hdc)                                                                                       |
| ttui    | GpStatus | GdipGetDpiX(GpGraphics* graphics,REAL* dpi)                                                                                    |
| ttui    | GpStatus | GdipGetDpiY(GpGraphics* graphics,REAL* dpi)                                                                                    |
| titui   | GpStatus | GdipGetEmHeight(GDIPGpFontFamily* family,INT style,UINT16* EmHeight)                                                           |
| ttuitui | GpStatus | GdipGetEncoderParameterList(GpImage* image,GDIPCLSID* clsidEncoder,UINT size,EncoderParameters* buffer)                        |
| ttui    | GpStatus | GdipGetEncoderParameterListSize(GpImage* image,GDIPCLSID* clsidEncoder,UINT* size)                                             |
| ttui    | GpStatus | GdipGetFamily(GpFont* font,GpFontFamily* *family)                                                                              |
| twuhui  | GpStatus | GdipGetFamilyName(GDIPGpFontFamily* family,WCHAR* name[LF_FACESIZE],LANGID language)                                           |
| ttui    | GpStatus | GdipGetFontCollectionFamilyCount(GpFontCollection* fontCollection,INT numFound)                                                |
| tittui  | GpStatus | GdipGetFontCollectionFamilyList(GpFontCollection* fontCollection,INT numSought,GpFontFamily* gpfamilies[],INT* numFound)       |
| ttui    | GpStatus | GdipGetFontHeight(GDIPGpFont* font,GDIPGpGraphics* graphics,REAL height)                                                       |
| titui   | GpStatus | GdipGetFontHeightGivenDPI(GDIPGpFont* font,REAL dpi,REAL* height)                                                              |
| ttui    | GpStatus | GdipGetFontSize(GpFont* font,REAL* size)                                                                                       |
| ttui    | GpStatus | GdipGetFontStyle(GpFont* font,INT* style)                                                                                      |
| ttui    | GpStatus | GdipGetFontUnit(GpFont* font,Unit* unit)                                                                                       |
| tui     | GpStatus | GdipGetGenericFontFamilyMonospace(GpFontFamily* *nativeFamily)                                                                 |
| tui     | GpStatus | GdipGetGenericFontFamilySansSerif(GpFontFamily* *nativeFamily)                                                                 |
| tui     | GpStatus | GdipGetGenericFontFamilySerif(GpFontFamily* *nativeFamily)                                                                     |
| ttui    | GpStatus | GdipGetHatchBackgroundColor(GpHatch* brush,ARGB* backcol)                                                                      |
| ttui    | GpStatus | GdipGetHatchForegroundColor(GpHatch* brush,ARGB* forecol)                                                                      |
| ttui    | GpStatus | GdipGetHatchStyle(GpHatch* brush,GpHatchStyle* hatchstyle)                                                                     |
| ttui    | GpStatus | GdipGetHemlFromMetafile(GpMetafile* metafile,HENHMETAFILE* hEML)                                                               |
| tttui   | GpStatus | GdipGetImageAttributesAdjustedPalette(GpImageAttributes* imageAttr,ColorPalette* colorPalette,ColorAdjustType colorAdjustType) |
| tttui   | GpStatus | GdipGetImageBounds(GpImage* image,GpRectF* srcRect,GpUnit* srcUnit)                                                            |
| uiuitui | GpStatus | GdipGetImageDecoders(UINT numDecoders,UINT size,ImageCodecInfo* decoders)                                                      |
| ttui    | GpStatus | GdipGetImageDecodersSize(UINT* numDecoders,UINT* size)                                                                         |
| tttui   | GpStatus | GdipGetImageDimension(GpImage* image,REAL* width,REAL* height)                                                                 |
| uiuitui | GpStatus | GdipGetImageEncoders(UINT numEncoders,UINT size,ImageCodecInfo* encoders)                                                      |

|           |          |                                                                                                                                                 |
|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| ttui      | GpStatus | GdipGetImageEncodersSize (UINT* numEncoders,UINT* size)                                                                                         |
| ttui      | GpStatus | GdipGetImageFlags (GpImage* image,UINT* flags)                                                                                                  |
| ttui      | GpStatus | GdipGetImageGraphicsContext(GpImage* image,GpGraphics* *graphics)                                                                               |
| ttui      | GpStatus | GdipGetImageHeight (GpImage* image,UINT* height)                                                                                                |
| ttui      | GpStatus | GdipGetImageHorizontalResolution (GpImage* image,REAL* resolution)                                                                              |
| ttui      | GpStatus | GdipGetImagePalette (GpImage* image,ColorPalette* palette,INT size)                                                                             |
| ttui      | GpStatus | GdipGetImagePaletteSize(GpImage* image,INT* size)                                                                                               |
| ttui      | GpStatus | GdipGetImagePixelFormat(GpImage* image,PixelFormat* format)                                                                                     |
| ttui      | GpStatus | GdipGetImageRawFormat(GpImage* image,GUID* format)                                                                                              |
| tuiuittui | GpStatus | GdipGetImageThumbnail (GpImage* image,UINT thumbWidth,UINT thumbHeight,GpImage* *thumbImage,GetThumbnailImageAbort callback,VOID* callbackData) |
| ttui      | GpStatus | GdipGetImageType (GpImage* image,ImageType* type)                                                                                               |
| ttui      | GpStatus | GdipGetImageVerticalResolution(GpImage* image,REAL* resolution)                                                                                 |
| ttui      | GpStatus | GdipGetImageWidth(GpImage* image,UINT* width)                                                                                                   |
| ttui      | GpStatus | GdipGetInterpolationMode(GpGraphics* graphics,InterpolationMode* interpolationMode)                                                             |
| tttiii    | GpStatus | GdipGetLineBlend(GpLineGradient* brush,REAL* blend,REAL* positions,INT count)                                                                   |
| ttui      | GpStatus | GdipGetLineBlendCount(GpLineGradient* brush,INT* count)                                                                                         |
| ttui      | GpStatus | GdipGetLineColors (GpLineGradient* brush,ARGB* colors)                                                                                          |
| ttui      | GpStatus | GdipGetLineGammaCorrection(GpLineGradient* brush,BOOL* useGammaCorrection)                                                                      |
| tttiii    | GpStatus | GdipGetLinePresetBlend (GpLineGradient* brush,ARGB* blend,REAL* positions,INT count)                                                            |
| ttui      | GpStatus | GdipGetLinePresetBlendCount (GpLineGradient* brush,INT* count)                                                                                  |
| ttui      | GpStatus | GdipGetLineRect(GpLineGradient* brush,GpRectF* rect)                                                                                            |
| ttui      | GpStatus | GdipGetLineRect (GpLineGradient* brush,GpRect* rect)                                                                                            |
| titui     | GpStatus | GdipGetLineSpacing(GDIPGpFontFamily* family,INT style,UINT16* LineSpacing)                                                                      |
| ttui      | GpStatus | GdipGetLineTransform(GpLineGradient* brush,GpMatrix* matrix)                                                                                    |
| ttui      | GpStatus | GdipGetLineWrapMode(GpLineGradient* brush,GpWrapMode* wrapmode)                                                                                 |
| tttiii    | GpStatus | GdipGetLogFont (GpFont* font,GpGraphics* graphics,LOGFONTA* logfont)                                                                            |
| tttiii    | GpStatus | GdipGetLogFontA (GpFont* font,GpGraphics* graphics,LOGFONTA* logfontA)                                                                          |
| tttiii    | GpStatus | GdipGetLogFontW(GpFont* font,GpGraphics* graphics,LOGFONTW* logfontW)                                                                           |
| ttui      | GpStatus | GdipGetMatrixElements(GDIPGpMatrix* matrix,REAL* matrixOut)                                                                                     |

|        |          |                                                                                                                         |
|--------|----------|-------------------------------------------------------------------------------------------------------------------------|
| ttui   | GpStatus | GdipGetMetafileDownLevelRasterizationLimit (GDIPGpMetafile* metafile,UINT* metafileRasterizationLimitDpi)               |
| ttui   | GpStatus | GdipGetMetafileHeaderFromEmf (HENHMETAFILE hEmf,MetafileHeader* header)                                                 |
| ttui   | GpStatus | GdipGetMetafileHeaderFromFile (GDIPWCHAR* filename,MetafileHeader* header)                                              |
| ttui   | GpStatus | GdipGetMetafileHeaderFromMetafile (GpMetafile* metafile,MetafileHeader* header)                                         |
| ttui   | GpStatus | GdipGetMetafileHeaderFromStream (IStream* stream,MetafileHeader* header)                                                |
| tttui  | GpStatus | GdipGetMetafileHeaderFromWmf (HMETAFILE hWmf,GDIPWmfPlaceableFileHeader* wmfPlaceableFileHeader,MetafileHeader* header) |
| ttui   | GpStatus | GdipGetNearestColor (GpGraphics* graphics,ARGB* argb)                                                                   |
| ttui   | GpStatus | GdipGetPageScale (GpGraphics* graphics,REAL* scale)                                                                     |
| ttui   | GpStatus | GdipGetPageUnit (GpGraphics* graphics,GpUnit* unit)                                                                     |
| ttui   | GpStatus | GdipGetPathData (GpPath* path,GpPathData* pathData)                                                                     |
| ttui   | GpStatus | GdipGetPathFillMode (GpPath* path,GpFillMode* fillmode)                                                                 |
| tttiii | GpStatus | GdipGetPathGradientBlend (GpPathGradient* brush,REAL* blend,REAL* positions,INT count)                                  |
| ttui   | GpStatus | GdipGetPathGradientBlendCount (GpPathGradient* brush,INT* count)                                                        |
| ttui   | GpStatus | GdipGetPathGradientCenterColor (GpPathGradient* brush,ARGB* colors)                                                     |
| ttui   | GpStatus | GdipGetPathGradientCenterPoint (GpPathGradient* brush,GpPointF* point)                                                  |
| ttui   | GpStatus | GdipGetPathGradientCenterPointInt (GpPathGradient* brush,GpPoint* point)                                                |
| tttiii | GpStatus | GdipGetPathGradientFocusScale (GpPathGradient* brush,REAL* xScale,REAL* yScale)                                         |
| ttui   | GpStatus | GdipGetPathGradientGammaCorrection (GpPathGradient* brush,BOOL* useGammaCorrection)                                     |
| ttui   | GpStatus | GdipGetPathGradientPath (GpPathGradient* brush,GpPath* path)                                                            |
| ttui   | GpStatus | GdipGetPathGradientPointCount (GpPathGradient* brush,INT* count)                                                        |
| tttiii | GpStatus | GdipGetPathGradientPresetBlend (GpPathGradient* brush,ARGB* blend,REAL* positions,INT count)                            |
| ttui   | GpStatus | GdipGetPathGradientPresetBlendCount (GpPathGradient* brush,INT* count)                                                  |
| ttui   | GpStatus | GdipGetPathGradientRect (GpPathGradient* brush,GpRectF* rect)                                                           |
| ttui   | GpStatus | GdipGetPathGradientRectInt (GpPathGradient* brush,GpRect* rect)                                                         |
| ttui   | GpStatus | GdipGetPathGradientSurroundColorCount (GpPathGradient* brush,INT* count)                                                |
| tttiii | GpStatus | GdipGetPathGradientSurroundColorsWithCount (GpPathGradient* brush,ARGB* color,INT* count)                               |
|        |          |                                                                                                                         |

|        |          |                                                                                          |
|--------|----------|------------------------------------------------------------------------------------------|
| ttui   | GpStatus | GdipGetPathGradientTransform(GpPathGradient* brush,GpMatrix* matrix)                     |
| ttui   | GpStatus | GdipGetPathGradientWrapMode(GpPathGradient* brush,GpWrapMode* wrapmode)                  |
| ttui   | GpStatus | GdipGetPathLastPoint(GpPath* path,GpPointF* lastPoint)                                   |
| ttui   | GpStatus | GdipGetPathPoints(GpPath*,GpPointF* points,INT count)                                    |
| ttui   | GpStatus | GdipGetPathPointsI(GpPath*,GpPoint* points,INT count)                                    |
| ttui   | GpStatus | GdipGetPathTypes(GpPath* path,BYTE* types,INT count)                                     |
| tttui  | GpStatus | GdipGetPathWorldBounds(GpPath* path,GpRectF* bounds,GDIPGpMatrix* matrix,GDIPGpPen* pen) |
| tttui  | GpStatus | GdipGetPathWorldBoundsI(GpPath* path,GpRect* bounds,GDIPGpMatrix* matrix,GDIPGpPen* pen) |
| ttui   | GpStatus | GdipGetPenBrushFill(GpPen* pen,GpBrush* *brush)                                          |
| ttui   | GpStatus | GdipGetPenColor(GpPen* pen,ARGB* argb)                                                   |
| ttui   | GpStatus | GdipGetPenCompoundArray(GpPen* pen,REAL* dash,INT count)                                 |
| ttui   | GpStatus | GdipGetPenCompoundCount(GpPen* pen,INT* count)                                           |
| ttui   | GpStatus | GdipGetPenCustomEndCap(GpPen* pen,GpCustomLineCap** customCap)                           |
| ttui   | GpStatus | GdipGetPenCustomStartCap(GpPen* pen,GpCustomLineCap** customCap)                         |
| ttui   | GpStatus | GdipGetPenDashArray(GpPen* pen,REAL* dash,INT count)                                     |
| ttui   | GpStatus | GdipGetPenDashCap197819(GpPen* pen,GpDashCap* dashCap)                                   |
| ttui   | GpStatus | GdipGetPenDashCount(GpPen* pen,INT* count)                                               |
| ttui   | GpStatus | GdipGetPenDashOffset(GpPen* pen,REAL* offset)                                            |
| ttui   | GpStatus | GdipGetPenDashStyle(GpPen* pen,GpDashStyle* dashstyle)                                   |
| ttui   | GpStatus | GdipGetPenEndCap(GpPen* pen,GpLineCap* endCap)                                           |
| ttui   | GpStatus | GdipGetPenFillType(GpPen* pen,GpPenType* type)                                           |
| ttui   | GpStatus | GdipGetPenLineJoin(GpPen* pen,GpLineJoin* lineJoin)                                      |
| ttui   | GpStatus | GdipGetPenMiterLimit(GpPen* pen,REAL* miterLimit)                                        |
| ttui   | GpStatus | GdipGetPenMode(GpPen* pen,GpPenAlignment* penMode)                                       |
| ttui   | GpStatus | GdipGetPenStartCap(GpPen* pen,GpLineCap* startCap)                                       |
| ttui   | GpStatus | GdipGetPenTransform(GpPen* pen,GpMatrix* matrix)                                         |
| ttui   | GpStatus | GdipGetPenUnit(GpPen* pen,GpUnit* unit)                                                  |
| ttui   | GpStatus | GdipGetPenWidth(GpPen* pen,REAL* width)                                                  |
| ttui   | GpStatus | GdipGetPixelOffsetMode(GpGraphics* graphics,PixelOffsetMode* pixelOffsetMode)            |
| ttui   | GpStatus | GdipGetPointCount(GpPath* path,INT* count)                                               |
| ttui   | GpStatus | GdipGetPropertyCount(GpImage* image,UINT* numOfProperty)                                 |
| tuitui | GpStatus | GdipGetPropertyIdList(GpImage* image,UINT numOfProperty,PROPID*                          |

|         |          |                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------|
| tuiitui | GpStatus | GdipGetPropertyItem(GpImage* image,PROPID propId,UINT propSize,PropertyItem* buffer)                                |
| tuitui  | GpStatus | GdipGetPropertyItemSize(GpImage* image,PROPID propId,UINT* size)                                                    |
| tttui   | GpStatus | GdipGetPropertySize(GpImage* image,UINT* totalBufferSize,UINT* numProperties)                                       |
| tttui   | GpStatus | GdipGetRegionBounds(GpRegion* region,GpGraphics* graphics,GpRectF rect)                                             |
| tttui   | GpStatus | GdipGetRegionBoundsI(GpRegion* region,GpGraphics* graphics,GpRect rect)                                             |
| ttuitui | GpStatus | GdipGetRegionData(GpRegion* region,BYTE* buffer,UINT bufferSize,UInt sizeFilled)                                    |
| ttui    | GpStatus | GdipGetRegionDataSize(GpRegion* region,UINT* bufferSize)                                                            |
| tttui   | GpStatus | GdipGetRegionHRgn(GpRegion* region,GpGraphics* graphics,HRGN* hrgn)                                                 |
| ttttui  | GpStatus | GdipGetRegionScans(GpRegion* region,GpRectF* rects,INT* count,GpMatrix* matrix)                                     |
| tttui   | GpStatus | GdipGetRegionScansCount(GpRegion* region,UINT* count,GpMatrix* matrix)                                              |
| ttttui  | GpStatus | GdipGetRegionScansI(GpRegion* region,GpRect* rects,INT* count,GpMatrix* matrix)                                     |
| tttui   | GpStatus | GdipGetRenderingOrigin(GpGraphics* graphics,INT* x,INT* y)                                                          |
| ttui    | GpStatus | GdipGetSmoothingMode(GpGraphics* graphics,SmoothingMode* smoothingMode)                                             |
| ttui    | GpStatus | GdipGetSolidFillColor(GpSolidFill* brush,ARGB* color)                                                               |
| ttui    | GpStatus | GdipGetStringFormatAlign(GDIPGpStringFormat* format,StringAlignment* align)                                         |
| tttui   | GpStatus | GdipGetStringFormatDigitSubstitution(GDIPGpStringFormat* format,LANGID* language,StringDigitSubstitute* substitute) |
| ttui    | GpStatus | GdipGetStringFormatFlags(GDIPGpStringFormat* format,INT* flags)                                                     |
| ttui    | GpStatus | GdipGetStringFormatHotkeyPrefix(GDIPGpStringFormat* format,INT* hotkeyPrefix)                                       |
| ttui    | GpStatus | GdipGetStringFormatLineAlign(GDIPGpStringFormat* format,StringAlignment* align)                                     |
| ttui    | GpStatus | GdipGetStringFormatMeasurableCharacterRangeCount(GDIPGpStringFormat* format,INT* count)                             |
| ttui    | GpStatus | GdipGetStringFormatTabStopCount(GDIPGpStringFormat* format,INT* count)                                              |
| tittui  | GpStatus | GdipGetStringFormatTabStops(GDIPGpStringFormat* format,INT count,REAL* firstTabOffset,REAL* tabStops)               |
| ttui    | GpStatus | GdipGetStringFormatTrimming(GDIPGpStringFormat* format,StringTrimming* trimming)                                    |
|         |          |                                                                                                                     |

|          |          |                                                                                                        |
|----------|----------|--------------------------------------------------------------------------------------------------------|
| ttui     | GpStatus | GdipGetTextRenderingHint(GpGraphics* graphics,TextRenderingHint* m                                     |
| ttui     | GpStatus | GdipGetTextureImage(GpTexture* brush,GpImage* *image)                                                  |
| ttui     | GpStatus | GdipGetTextureTransform(GpTexture* brush,GpMatrix* matrix)                                             |
| ttui     | GpStatus | GdipGetTextureWrapMode(GpTexture* brush,GpWrapMode* wrapmode)                                          |
| ttui     | GpStatus | GdipGetVisibleClipBounds(GpGraphics* graphics,GpRectF* rect)                                           |
| ttui     | GpStatus | GdipGetVisibleClipBoundsI(GpGraphics* graphics,GpRect* rect)                                           |
| ttui     | GpStatus | GdipGetWorldTransform(GpGraphics* graphics,GpMatrix* matrix)                                           |
| tuiui    | GpStatus | GdipGraphicsClear(GpGraphics* graphics,ARGB color)                                                     |
| tui      | GpStatus | GdipImageForceValidation(GpImage* image)                                                               |
| tttui    | GpStatus | GdipImageGetFrameCount(GpImage* image,GDIPGUID* dimensionID,UINT* count)                               |
| ttui     | GpStatus | GdipImageGetFrameDimensionsCount(GpImage* image,UINT* count)                                           |
| ttuiui   | GpStatus | GdipImageGetFrameDimensionsList(GpImage* image,GUID* dimensionIDs,UINT count)                          |
| tuiui    | GpStatus | GdipImageRotateFlip(GpImage* image,RotateFlipType rfType)                                              |
| ttuiui   | GpStatus | GdipImageSelectActiveFrame(GpImage* image,GDIPGUID* dimensionID,UINT frameIndex)                       |
| tui      | GpStatus | GdipInvertMatrix(GpMatrix* matrix)                                                                     |
| ttui     | GpStatus | GdipIsClipEmpty(GpGraphics* graphics,BOOL* result)                                                     |
| tttui    | GpStatus | GdipIsEmptyRegion(GpRegion* region,GpGraphics* graphics,BOOL* res                                      |
| ttttui   | GpStatus | GdipIsEqualRegion(GpRegion* region,GpRegion* region2,GpGraphics* graphics,BOOL* result)                |
| tttui    | GpStatus | GdipIsInfiniteRegion(GpRegion* region,GpGraphics* graphics,BOOL* re                                    |
| tttui    | GpStatus | GdipIsMatrixEqual(GDIPGpMatrix* matrix,GDIPGpMatrix* matrix2,BO result)                                |
| ttui     | GpStatus | GdipIsMatrixIdentity(GDIPGpMatrix* matrix,BOOL* result)                                                |
| ttui     | GpStatus | GdipIsMatrixInvertible(GDIPGpMatrix* matrix,BOOL* result)                                              |
| tiitttui | GpStatus | GdipIsOutlineVisiblePathPoint(GpPath* path,REAL x,REAL y,GpPen* pen,GpGraphics* graphics,BOOL* result) |
| tiitttui | GpStatus | GdipIsOutlineVisiblePathPointI(GpPath* path,INT x,INT y,GpPen* pen,GpGraphics* graphics,BOOL* result)  |
| ttui     | GpStatus | GdipIsVisibleClipEmpty(GpGraphics* graphics,BOOL* result)                                              |
| tiittui  | GpStatus | GdipIsVisiblePathPoint(GpPath* path,REAL x,REAL y,GpGraphics* graphics,BOOL* result)                   |
| tiittui  | GpStatus | GdipIsVisiblePathPointI(GpPath* path,INT x,INT y,GpGraphics* graphics,BOOL* result)                    |
| tiitui   | GpStatus | GdipIsVisiblePoint(GpGraphics* graphics,REAL x,REAL y,BOOL* result)                                    |
| tiitui   | GpStatus | GdipIsVisiblePointI(GpGraphics* graphics,INT x,INT y,BOOL* result)                                     |

|             |          |                                                                                                                                                                                                         |
|-------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiiiiitui   | GpStatus | GdiplVisibleRect(GpGraphics* graphics,REAL x,REAL y,REAL width,REAL height,BOOL* result)                                                                                                                |
| tiiiiitui   | GpStatus | GdiplVisibleRect(GpGraphics* graphics,INT x,INT y,INT width,INT height,BOOL* result)                                                                                                                    |
| tiittui     | GpStatus | GdiplVisibleRegionPoint(GpRegion* region,REAL x,REAL y,GpGraphics* graphics,BOOL* result)                                                                                                               |
| tiittui     | GpStatus | GdiplVisibleRegionPoint(GpRegion* region,INT x,INT y,GpGraphics* graphics,BOOL* result)                                                                                                                 |
| tiiiittui   | GpStatus | GdiplVisibleRegionRect(GpRegion* region,REAL x,REAL y,REAL width,REAL height,GpGraphics* graphics,BOOL* result)                                                                                         |
| tiiiittui   | GpStatus | GdiplVisibleRegionRect(GpRegion* region,INT x,INT y,INT width,INT height,GpGraphics* graphics,BOOL* result)                                                                                             |
| ttui        | GpStatus | GdiplLoadImageFromFile(GDIPWCHAR* filename,GpImage* *image)                                                                                                                                             |
| ttui        | GpStatus | GdiplLoadImageFromFileICM(GDIPWCHAR* filename,GpImage* *image)                                                                                                                                          |
| ttui        | GpStatus | GdiplLoadImageFromStream(IStream* stream,GpImage* *image)                                                                                                                                               |
| ttui        | GpStatus | GdiplLoadImageFromStreamICM(IStream* stream,GpImage* *image)                                                                                                                                            |
| tui         | GpStatus | GdiplusNotificationHook(ULONG_PTR* token)                                                                                                                                                               |
| ti          | VOID     | GdiplusNotificationUnhook(ULONG_PTR token)                                                                                                                                                              |
| ti          | VOID     | GdiplusShutdown(__in ULONG_PTR token)                                                                                                                                                                   |
| ttui        | Status   | GdiplusStartup(__out ULONG_PTR token *token, __in const GdiplusStartupInput *input, __out GdiplusStartupOutput *output)                                                                                 |
| ttituititui | GpStatus | GdiplMeasureCharacterRanges(GpGraphics* graphics,GDIPWCHAR* string,INT length,GDIPGpFont* font,GDIPRectF &layoutRect,GDIPGpStringFormat* stringFormat,INT regionCount,GpRegion* *regions)               |
| ttittitui   | GpStatus | GdiplMeasureDriverString(GpGraphics* graphics,GDIPUINT16* text,INT length,GDIPGpFont* font,GDIPPointF* positions,INT flags,GDIPGpMatrix* matrix,RectF* boundingBox)                                     |
| ttittttitui | GpStatus | GdiplMeasureString(GpGraphics* graphics,GDIPWCHAR* string,INT length,GDIPGpFont* font,GDIPRectF* layoutRect,GDIPGpStringFormat* stringFormat,RectF* boundingBox,INT* codepointsFitted,INT* linesFilled) |
| ttuiui      | GpStatus | GdiplMultiplyLineTransform(GpLineGradient* brush,GDIPGpMatrix* matrix,GpMatrixOrder order)                                                                                                              |
| ttuiui      | GpStatus | GdiplMultiplyMatrix(GpMatrix* matrix,GpMatrix* matrix2,GpMatrixOrder order)                                                                                                                             |
| ttuiui      | GpStatus | GdiplMultiplyPathGradientTransform(GpPathGradient* brush,GDIPGpMatrix* matrix,GpMatrixOrder order)                                                                                                      |
| ttuiui      | GpStatus | GdiplMultiplyPenTransform(GpPen* pen,GDIPGpMatrix* matrix,GpMatrixOrder order)                                                                                                                          |
| ttuiui      | GpStatus | GdiplMultiplyTextureTransform(GpTexture* brush,GDIPGpMatrix* matrix,GpMatrixOrder order)                                                                                                                |

|           |          |                                                                                                                                                                               |
|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiui    | GpStatus | GdipMultiplyWorldTransform(GpGraphics* graphics,GDIPGpMatrix* matrix,GpMatrixOrder order)                                                                                     |
| tui       | GpStatus | GdipNewInstalledFontCollection(GpFontCollection** fontCollection)                                                                                                             |
| tui       | GpStatus | GdipNewPrivateFontCollection(GpFontCollection** fontCollection)                                                                                                               |
| tttiiiui  | GpStatus | GdipPathIterCopyData(GpPathIterator* iterator,INT* resultCount,GpPoint points,BYTE* types,INT startIndex,INT endIndex)                                                        |
| tttiiiui  | GpStatus | GdipPathIterEnumerate(GpPathIterator* iterator,INT* resultCount,GpPoint points,BYTE* types,INT count)                                                                         |
| ttui      | GpStatus | GdipPathIterGetCount(GpPathIterator* iterator,INT* count)                                                                                                                     |
| ttui      | GpStatus | GdipPathIterGetSubpathCount(GpPathIterator* iterator,INT* count)                                                                                                              |
| ttui      | GpStatus | GdipPathIterHasCurve(GpPathIterator* iterator,BOOL* hasCurve)                                                                                                                 |
| ttui      | GpStatus | GdipPathIterIsValid(GpPathIterator* iterator,BOOL* valid)                                                                                                                     |
| ttttui    | GpStatus | GdipPathIterNextMarker(GpPathIterator* iterator,INT* resultCount,INT* startIndex,INT* endIndex)                                                                               |
| tttui     | GpStatus | GdipPathIterNextMarkerPath(GpPathIterator* iterator,INT* resultCount,GpPath* path)                                                                                            |
| tttttui   | GpStatus | GdipPathIterNextPathType(GpPathIterator* iterator,INT* resultCount,BYTE* pathType,INT* startIndex,INT* endIndex)                                                              |
| tttttui   | GpStatus | GdipPathIterNextSubpath(GpPathIterator* iterator,INT* resultCount,INT* startIndex,INT* endIndex,BOOL* isClosed)                                                               |
| ttttui    | GpStatus | GdipPathIterNextSubpathPath(GpPathIterator* iterator,INT* resultCount,GpPath* path,BOOL* isClosed)                                                                            |
| tui       | GpStatus | GdipPathIterRewind(GpPathIterator* iterator)                                                                                                                                  |
| ttuiuitui | GpStatus | GdipPlayMetafileRecord(GDIPGpMetafile* metafile,EmfPlusRecordType recordType,UINT flags,UINT dataSize,GDIPBYTE* data)                                                         |
| ttui      | GpStatus | GdipPrivateAddFontFile(GpFontCollection* fontCollection,GDIPWCHAR* filename)                                                                                                  |
| ttui      | GpStatus | GdipPrivateAddMemoryFont(GpFontCollection* fontCollection,GDIPvoid* memory,INT length)                                                                                        |
| tttuitui  | GpStatus | GdipRecordMetafile(HDC referenceHdc,EmfType type,GDIPGpRectF* frameRect,MetafileFrameUnit frameUnit,GDIPWCHAR* description,GpMetafile** metafile)                             |
| ttttuitui | GpStatus | GdipRecordMetafileFileName(GDIPWCHAR* fileName,HDC referenceHdc,EmfType type,GDIPGpRectF* frameRect,MetafileFrameUnit frameUnit,GDIPWCHAR* description,GpMetafile** metafile) |
| ttttuitui | GpStatus | GdipRecordMetafileFileName1(GDIPWCHAR* fileName,HDC referenceHdc,EmfType type,GDIPGpRect* frameRect,MetafileFrameUnit frameUnit,GDIPWCHAR* description,GpMetafile** metafile) |
| tttuitui  | GpStatus | GdipRecordMetafile1(HDC referenceHdc,EmfType type,GDIPGpRect* frameRect,MetafileFrameUnit frameUnit,GDIPWCHAR* description,GpMetafile** metafile)                             |

|           |          |                                                                                                                                                                         |
|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tttuittui | GpStatus | GdipRecordMetafileStream(IStream* stream,HDC referenceHdc,EmfType type,GDIPGpRectF* frameRect,MetafileFrameUnit frameUnit,GDIPWCHAR description,GpMetafile* * metafile) |
| tttuittui | GpStatus | GdipRecordMetafileStream(IStream* stream,HDC referenceHdc,EmfType type,GDIPGpRect* frameRect,MetafileFrameUnit frameUnit,GDIPWCHAR description,GpMetafile* * metafile)  |
| ttui      | GpStatus | GdipReleaseDC(GpGraphics* graphics,HDC hdc)                                                                                                                             |
| tuiui     | GpStatus | GdipRemovePropertyItem(GpImage* image,PROPID propId)                                                                                                                    |
| tui       | GpStatus | GdipResetClip(GpGraphics* graphics)                                                                                                                                     |
| ttui      | GpStatus | GdipResetImageAttributes(GpImageAttributes* imageattr,ColorAdjustType type)                                                                                             |
| tui       | GpStatus | GdipResetLineTransform(GpLineGradient* brush)                                                                                                                           |
| tui       | GpStatus | GdipResetPageTransform(GpGraphics* graphics)                                                                                                                            |
| tui       | GpStatus | GdipResetPath(GpPath* path)                                                                                                                                             |
| tui       | GpStatus | GdipResetPathGradientTransform(GpPathGradient* brush)                                                                                                                   |
| tui       | GpStatus | GdipResetPenTransform(GpPen* pen)                                                                                                                                       |
| tui       | GpStatus | GdipResetTextureTransform(GpTexture* brush)                                                                                                                             |
| tui       | GpStatus | GdipResetWorldTransform(GpGraphics* graphics)                                                                                                                           |
| tuiui     | GpStatus | GdipRestoreGraphics(GpGraphics* graphics,GraphicsState state)                                                                                                           |
| tui       | GpStatus | GdipReversePath(GpPath* path)                                                                                                                                           |
| tiuiui    | GpStatus | GdipRotateLineTransform(GpLineGradient* brush,REAL angle,GpMatrixOrder order)                                                                                           |
| tiuiui    | GpStatus | GdipRotateMatrix(GpMatrix* matrix,REAL angle,GpMatrixOrder order)                                                                                                       |
| tiuiui    | GpStatus | GdipRotatePathGradientTransform(GpPathGradient* brush,REAL angle,GpMatrixOrder order)                                                                                   |
| tiuiui    | GpStatus | GdipRotatePenTransform(GpPen* pen,REAL angle,GpMatrixOrder order)                                                                                                       |
| tiuiui    | GpStatus | GdipRotateTextureTransform(GpTexture* brush,REAL angle,GpMatrixOrder order)                                                                                             |
| tiuiui    | GpStatus | GdipRotateWorldTransform(GpGraphics* graphics,REAL angle,GpMatrixOrder order)                                                                                           |
| ttui      | GpStatus | GdipSaveAdd(GpImage* image,GDIPEncoderParameters* encoderParams)                                                                                                        |
| tttui     | GpStatus | GdipSaveAddImage(GpImage* image,GpImage* newImage,GDIPEncoderParameters* encoderParams)                                                                                 |
| ttui      | GpStatus | GdipSaveGraphics(GpGraphics* graphics,GraphicsState* state)                                                                                                             |
| ttttui    | GpStatus | GdipSaveImageToFile(GpImage* image,GDIPWCHAR* filename,GDIPCLSID* clsidEncoder,GDIPEncoderParameters* encoderParams)                                                    |
| ttttui    | GpStatus | GdipSaveImageToStream(GpImage* image,IStream* stream,GDIPCLSID* clsidEncoder,GDIPEncoderParameters* encoderParams)                                                      |

|          |          |                                                                                                            |
|----------|----------|------------------------------------------------------------------------------------------------------------|
| tiiuii   | GpStatus | <i>GdipScaleLineTransform</i> (GpLineGradient* brush,REAL sx,REAL sy,GpMatrixOrder order)                  |
| tiiuii   | GpStatus | <i>GdipScaleMatrix</i> (GpMatrix* matrix,REAL scaleX,REAL scaleY,GpMatrixOrder order)                      |
| tiiuii   | GpStatus | <i>GdipScalePathGradientTransform</i> (GpPathGradient* brush,REAL sx,REAL sy,GpMatrixOrder order)          |
| tiiuii   | GpStatus | <i>GdipScalePenTransform</i> (GpPen* pen,REAL sx,REAL sy,GpMatrixOrder order)                              |
| tiiuii   | GpStatus | <i>GdipScaleTextureTransform</i> (GpTexture* brush,REAL sx,REAL sy,GpMatrixOrder order)                    |
| tiiuii   | GpStatus | <i>GdipScaleWorldTransform</i> (GpGraphics* graphics,REAL sx,REAL sy,GpMatrixOrder order)                  |
| tiui     | GpStatus | <i>GdipSetAdjustableArrowCapFillState</i> (GpAdjustableArrowCap* cap,BOOL fillState)                       |
| tiui     | GpStatus | <i>GdipSetAdjustableArrowCapHeight</i> (GpAdjustableArrowCap* cap,REAL height)                             |
| tiui     | GpStatus | <i>GdipSetAdjustableArrowCapMiddleInset</i> (GpAdjustableArrowCap* cap,REAL middleInset)                   |
| tiui     | GpStatus | <i>GdipSetAdjustableArrowCapWidth</i> (GpAdjustableArrowCap* cap,REAL width)                               |
| ttui     | GpStatus | <i>GdipSetClipGraphics</i> (GpGraphics* graphics,GpGraphics* srcgraphics,CombineMode combineMode)          |
| ttui     | GpStatus | <i>GdipSetClipHrgn</i> (GpGraphics* graphics,HRGN hRgn,CombineMode combineMode)                            |
| ttui     | GpStatus | <i>GdipSetClipPath</i> (GpGraphics* graphics,GpPath* path,CombineMode combineMode)                         |
| tiiiitui | GpStatus | <i>GdipSetClipRect</i> (GpGraphics* graphics,REAL x,REAL y,REAL width,REAL height,CombineMode combineMode) |
| tiiiitui | GpStatus | <i>GdipSetClipRectI</i> (GpGraphics* graphics,INT x,INT y,INT width,INT height,CombineMode combineMode)    |
| ttui     | GpStatus | <i>GdipSetClipRegion</i> (GpGraphics* graphics,GpRegion* region,CombineMode combineMode)                   |
| ttui     | GpStatus | <i>GdipSetCompositingMode</i> (GpGraphics* graphics,CompositingMode compositingMode)                       |
| ttui     | GpStatus | <i>GdipSetCompositingQuality</i> (GpGraphics* graphics,CompositingQuality compositingQuality)              |
| tuiui    | GpStatus | <i>GdipSetCustomLineCapBaseCap</i> (GpCustomLineCap* customCap,GpLineCap baseCap)                          |
| tiui     | GpStatus | <i>GdipSetCustomLineCapBaseInset</i> (GpCustomLineCap* customCap,REAL inset)                               |
| tuiuii   | GpStatus | <i>GdipSetCustomLineCapStrokeCaps</i> (GpCustomLineCap* customCap,GpLineCap startCap,GpLineCap endCap)     |

|          |          |                                                                                                                                                                                       |
|----------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiui    | GpStatus | GdipSetCustomLineCapStrokeJoin (GpCustomLineCap* customCap,GpLineJoin lineJoin)                                                                                                       |
| tiui     | GpStatus | GdipSetCustomLineCapWidthScale(GpCustomLineCap* customCap,REAL widthScale)                                                                                                            |
| tui      | GpStatus | GdipSetEmpty (GpRegion* region)                                                                                                                                                       |
| tiui     | GpStatus | GdipSetImageAttributesCachedBackground(GpImageAttributes* imageattr,BOOL enableFlag)                                                                                                  |
| ttuiuiui | GpStatus | GdipSetImageAttributesColorKeys (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,ARGB colorLow,ARGB colorHigh)                                                      |
| ttittui  | GpStatus | GdipSetImageAttributesColorMatrix (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,GDIPColorMatrix* colorMatrix,GDIPColorMatrix* grayMatrix,ColorMatrixFlags flags) |
| ttiii    | GpStatus | GdipSetImageAttributesGamma (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,REAL gamma)                                                                            |
| ttui     | GpStatus | GdipSetImageAttributesNoOp(GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag)                                                                                         |
| ttitui   | GpStatus | GdipSetImageAttributesOutputChannel (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,ColorChannelFlags channelFlags)                                                |
| ttitui   | GpStatus | GdipSetImageAttributesOutputChannelColorProfile (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,GDIPWCHAR* colorProfileFilename)                                   |
| ttiuitui | GpStatus | GdipSetImageAttributesRemapTable (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,UINT mapSize,GDIPColorMap* map)                                                   |
| ttiii    | GpStatus | GdipSetImageAttributesThreshold (GpImageAttributes* imageattr,ColorAdjustType type,BOOL enableFlag,REAL threshold)                                                                    |
| ttui     | GpStatus | GdipSetImageAttributesToIdentity (GpImageAttributes* imageattr,ColorAdjustType type)                                                                                                  |
| tuiiii   | GpStatus | GdipSetImageAttributesWrapMode (GpImageAttributes* imageAttr,WrapMode wrap,ARGB argb,BOOL clamp)                                                                                      |
| ttui     | GpStatus | GdipSetImagePalette (GpImage* image,GDIPColorPalette* palette)                                                                                                                        |
| tui      | GpStatus | GdipSetInfinite (GpRegion* region)                                                                                                                                                    |
| tuiui    | GpStatus | GdipSetInterpolationMode (GpGraphics* graphics,InterpolationMode interpolationMode)                                                                                                   |
| tttiii   | GpStatus | GdipSetLineBlend (GpLineGradient* brush,GDIPREAL* blend,GDIPREAL positions,INT count)                                                                                                 |
| tuiuiui  | GpStatus | GdipSetLineColors (GpLineGradient* brush,ARGB color1,ARGB color2)                                                                                                                     |
| tiui     | GpStatus | GdipSetLineGammaCorrection (GpLineGradient* brush,BOOL useGammaCorrection)                                                                                                            |
| ttiii    | GpStatus | GdipSetLineLinearBlend (GpLineGradient* brush,REAL focus,REAL scale)                                                                                                                  |

|          |          |                                                                                                     |
|----------|----------|-----------------------------------------------------------------------------------------------------|
| tttiii   | GpStatus | GdipSetLinePresetBlend(GpLineGradient* brush,GDIPARGB* blend,GDIPREAL* positions,INT count)         |
| tiii     | GpStatus | GdipSetLineSigmaBlend(GpLineGradient* brush,REAL focus,REAL scale)                                  |
| ttui     | GpStatus | GdipSetLineTransform(GpLineGradient* brush,GDIPGpMatrix* matrix)                                    |
| tuiii    | GpStatus | GdipSetLineWrapMode(GpLineGradient* brush,GpWrapMode wrapmode)                                      |
| tiiiiiii | GpStatus | GdipSetMatrixElement(GpMatrix* matrix,REAL m11,REAL m12,REAL m21,REAL m22,REAL dx,REAL dy)          |
| tuiii    | GpStatus | GdipSetMetafileDownLevelRasterizationLimit(GpMetafile* metafile,UINT metafileRasterizationLimitDpi) |
| tiii     | GpStatus | GdipSetPageScale(GpGraphics* graphics,REAL scale)                                                   |
| tuiii    | GpStatus | GdipSetPageUnit(GpGraphics* graphics,GpUnit unit)                                                   |
| tuiii    | GpStatus | GdipSetPathFillMode(GpPath* path,GpFillMode fillmode)                                               |
| tttiii   | GpStatus | GdipSetPathGradientBlend(GpPathGradient* brush,GDIPREAL* blend,GDIPREAL* positions,INT count)       |
| tuiii    | GpStatus | GdipSetPathGradientCenterColor(GpPathGradient* brush,ARGB colors)                                   |
| ttui     | GpStatus | GdipSetPathGradientCenterPoint(GpPathGradient* brush,GDIPGpPointF* points)                          |
| ttui     | GpStatus | GdipSetPathGradientCenterPointf(GpPathGradient* brush,GDIPGpPoint* points)                          |
| tiii     | GpStatus | GdipSetPathGradientFocusScale(GpPathGradient* brush,REAL xScale,REAL yScale)                        |
| tiii     | GpStatus | GdipSetPathGradientGammaCorrection(GpPathGradient* brush,BOOL useGammaCorrection)                   |
| tiii     | GpStatus | GdipSetPathGradientLinearBlend(GpPathGradient* brush,REAL focus,REAL scale)                         |
| ttui     | GpStatus | GdipSetPathGradientPath(GpPathGradient* brush,GDIPGpPath* path)                                     |
| tttiii   | GpStatus | GdipSetPathGradientPresetBlend(GpPathGradient* brush,GDIPARGB* blend,GDIPREAL* positions,INT count) |
| tiii     | GpStatus | GdipSetPathGradientSigmaBlend(GpPathGradient* brush,REAL focus,REAL scale)                          |
| tttiii   | GpStatus | GdipSetPathGradientSurroundColorsWithCount(GpPathGradient* brush,GDIPARGB* color,INT* count)        |
| ttui     | GpStatus | GdipSetPathGradientTransform(GpPathGradient* brush,GpMatrix* matrix)                                |
| tuiii    | GpStatus | GdipSetPathGradientWrapMode(GpPathGradient* brush,GpWrapMode wrapmode)                              |
| tui      | GpStatus | GdipSetPathMarker(GpPath* path)                                                                     |
| ttui     | GpStatus | GdipSetPenBrushFill(GpPen* pen,GpBrush* brush)                                                      |
| tuiii    | GpStatus | GdipSetPenColor(GpPen* pen,ARGB argb)                                                               |
| ttiii    | GpStatus | GdipSetPenCompoundArray(GpPen* pen,GDIPREAL* dash,INT count)                                        |

|           |          |                                                                                                                |
|-----------|----------|----------------------------------------------------------------------------------------------------------------|
| ttui      | GpStatus | GdipSetPenCustomEndCap(GpPen* pen,GpCustomLineCap* customCap)                                                  |
| ttui      | GpStatus | GdipSetPenCustomStartCap(GpPen* pen,GpCustomLineCap* customCap)                                                |
| ttiui     | GpStatus | GdipSetPenDashArray(GpPen* pen,GDIPREAL* dash,INT count)                                                       |
| tuiui     | GpStatus | GdipSetPenDashCap197819(GpPen* pen,GpDashCap dashCap)                                                          |
| tiui      | GpStatus | GdipSetPenDashOffset(GpPen* pen,REAL offset)                                                                   |
| tuiui     | GpStatus | GdipSetPenDashStyle(GpPen* pen,GpDashStyle dashstyle)                                                          |
| tuiui     | GpStatus | GdipSetPenEndCap(GpPen* pen,GpLineCap endCap)                                                                  |
| tuiuiuiui | GpStatus | GdipSetPenLineCap197819(GpPen* pen,GpLineCap startCap,GpLineCap endCap,GpDashCap dashCap)                      |
| tuiui     | GpStatus | GdipSetPenLineJoin(GpPen* pen,GpLineJoin lineJoin)                                                             |
| tiui      | GpStatus | GdipSetPenMiterLimit(GpPen* pen,REAL miterLimit)                                                               |
| tuiui     | GpStatus | GdipSetPenMode(GpPen* pen,GpPenAlignment penMode)                                                              |
| tuiui     | GpStatus | GdipSetPenStartCap(GpPen* pen,GpLineCap startCap)                                                              |
| ttui      | GpStatus | GdipSetPenTransform(GpPen* pen,GpMatrix* matrix)                                                               |
| tuiui     | GpStatus | GdipSetPenUnit(GpPen* pen,GpUnit unit)                                                                         |
| tiui      | GpStatus | GdipSetPenWidth(GpPen* pen,REAL width)                                                                         |
| ttui      | GpStatus | GdipSetPixelOffsetMode(GpGraphics* graphics,PixelOffsetMode pixelOffsetMode)                                   |
| ttui      | GpStatus | GdipSetPropertyItem(GpImage* image,GDIPPropertyItem* item)                                                     |
| ttiui     | GpStatus | GdipSetRenderingOrigin(GpGraphics* graphics,INT x,INT y)                                                       |
| tuiui     | GpStatus | GdipSetSmoothingMode(GpGraphics* graphics,SmoothingMode smoothingMode)                                         |
| tuiui     | GpStatus | GdipSetSolidFillColor(GpSolidFill* brush,ARGB color)                                                           |
| tuiui     | GpStatus | GdipSetStringFormatAlign(GpStringFormat* format,StringAlignment align)                                         |
| tuhuiui   | GpStatus | GdipSetStringFormatDigitSubstitution(GpStringFormat* format,LANGID language,StringDigitSubstitute substitute)  |
| tiui      | GpStatus | GdipSetStringFormatFlags(GpStringFormat* format,INT flags)                                                     |
| tiui      | GpStatus | GdipSetStringFormatHotkeyPrefix(GpStringFormat* format,INT hotkeyPrefix)                                       |
| tuiui     | GpStatus | GdipSetStringFormatLineAlign(GpStringFormat* format,StringAlignment align)                                     |
| titui     | GpStatus | GdipSetStringFormatMeasurableCharacterRanges(GpStringFormat* format,INT rangeCount,GDIPCharacterRange* ranges) |
| tiitui    | GpStatus | GdipSetStringFormatTabStops(GpStringFormat* format,REAL firstTabOffset,INT count,GDIPREAL* tabStops)           |
| tuiui     | GpStatus | GdipSetStringFormatTrimming(GpStringFormat* format,StringTrimming trimming)                                    |
| tuiui     | GpStatus | GdipSetTextRenderingHint(GpGraphics* graphics,TextRenderingHint mode)                                          |

|            |          |                                                                                                                                     |
|------------|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| ttui       | GpStatus | GdipSetTextTransform(GpTexture* brush,GDIPGpMatrix* matrix)                                                                         |
| tuiui      | GpStatus | GdipSetTextWrapMode(GpTexture* brush,GpWrapMode wrapmode)                                                                           |
| ttui       | GpStatus | GdipSetWorldTransform(GpGraphics* graphics,GpMatrix* matrix)                                                                        |
| tiiuiui    | GpStatus | GdipShearMatrix(GpMatrix* matrix,REAL shearX,REAL shearY,GpMatrixOrder order)                                                       |
| tui        | GpStatus | GdipStartPathFigure(GpPath* path)                                                                                                   |
| tui        | GpStatus | GdipStringFormatGetGenericDefault(GpStringFormat* *format)                                                                          |
| tui        | GpStatus | GdipStringFormatGetGenericTypography(GpStringFormat* *format)                                                                       |
| uitui      | GpStatus | GdipTestControl(GpTestControlEnum control,void* param)                                                                              |
| ttiui      | GpStatus | GdipTransformMatrixPoints(GpMatrix* matrix,GpPointF* pts,INT count)                                                                 |
| ttiui      | GpStatus | GdipTransformMatrixPointsI(GpMatrix* matrix,GpPoint* pts,INT count)                                                                 |
| ttui       | GpStatus | GdipTransformPath(GpPath* path,GpMatrix* matrix)                                                                                    |
| tuiuitui   | GpStatus | GdipTransformPoints(GpGraphics* graphics,GpCoordinateSpace destSpace,GpCoordinateSpace srcSpace,GpPointF* points,INT count)         |
| tuiuitui   | GpStatus | GdipTransformPointsI(GpGraphics* graphics,GpCoordinateSpace destSpace,GpCoordinateSpace srcSpace,GpPoint* points,INT count)         |
| ttui       | GpStatus | GdipTransformRegion(GpRegion* region,GpMatrix* matrix)                                                                              |
| tiiui      | GpStatus | GdipTranslateClip(GpGraphics* graphics,REAL dx,REAL dy)                                                                             |
| tiiui      | GpStatus | GdipTranslateClipI(GpGraphics* graphics,INT dx,INT dy)                                                                              |
| tiiuiui    | GpStatus | GdipTranslateLineTransform(GpLineGradient* brush,REAL dx,REAL dy,GpMatrixOrder order)                                               |
| tiiuiui    | GpStatus | GdipTranslateMatrix(GpMatrix* matrix,REAL offsetX,REAL offsetY,GpMatrixOrder order)                                                 |
| tiiuiui    | GpStatus | GdipTranslatePathGradientTransform(GpPathGradient* brush,REAL dx,REAL dy,GpMatrixOrder order)                                       |
| tiiuiui    | GpStatus | GdipTranslatePenTransform(GpPen* pen,REAL dx,REAL dy,GpMatrixOrder order)                                                           |
| tiiui      | GpStatus | GdipTranslateRegion(GpRegion* region,REAL dx,REAL dy)                                                                               |
| tiiui      | GpStatus | GdipTranslateRegionI(GpRegion* region,INT dx,INT dy)                                                                                |
| tiiuiui    | GpStatus | GdipTranslateTextureTransform(GpTexture* brush,REAL dx,REAL dy,GpMatrixOrder order)                                                 |
| tiiuiui    | GpStatus | GdipTranslateWorldTransform(GpGraphics* graphics,REAL dx,REAL dy,GpMatrixOrder order)                                               |
| ttiui      | GpStatus | GdipVectorTransformMatrixPoints(GpMatrix* matrix,GpPointF* pts,INT count)                                                           |
| ttiui      | GpStatus | GdipVectorTransformMatrixPointsI(GpMatrix* matrix,GpPoint* pts,INT count)                                                           |
| ttiiiiuiui | GpStatus | GdipWarpPath(GpPath* path,GpMatrix* matrix,GDIPGpPointF* points,INT count,REAL srcx,REAL srcy,REAL srcwidth,REAL srcheight,WarpMode |

|        |          |                                                                              |
|--------|----------|------------------------------------------------------------------------------|
|        |          | warpMode,REAL flatness)                                                      |
| tttiii | GpStatus | GdipWidenPath (GpPath* nativePath,GpPen* pen,GpMatrix* matrix,REAL flatness) |
| ttiii  | GpStatus | GdipWindingModeOutline (GpPath* path,GpMatrix* matrix,REAL flatnes           |

## Glu32.dll

|          |                |                                                                                                                                                        |
|----------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti       | VOID           | gluBeginCurve(GLUnurbs *nobj)                                                                                                                          |
| ti       | VOID           | gluBeginPolygon(GLUtesselator *tess)                                                                                                                   |
| ti       | VOID           | gluBeginSurface(GLUnurbs *nobj)                                                                                                                        |
| ti       | VOID           | gluBeginTrim(GLUnurbs *nobj)                                                                                                                           |
| iiiiiti  | VOID           | gluBuild1DMipmaps(GLenum target, GLint components, GLint width, GLenum format, GLenum type, const void *data)                                          |
| iiiiiti  | VOID           | gluBuild2DMipmaps(GLenum target, GLint components, GLint width, GLint height, GLenum format, GLenum type, const void *data)                            |
| tdddi    | VOID           | gluCylinder(GLUquadric *qobj, GLdouble baseRadius, GLdouble topRadius, GLdouble height, GLint slices, GLint stacks)                                    |
| ti       | VOID           | gluDeleteNurbsRenderer(GLUnurbs *nobj)                                                                                                                 |
| ti       | VOID           | gluDeleteQuadric(GLUquadricObj *state)                                                                                                                 |
| ti       | VOID           | gluDeleteTess(GLUtesselator *tess)                                                                                                                     |
| tdddi    | VOID           | gluDisk(GLUquadric *qobj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops)                                                       |
| ti       | VOID           | gluEndCurve(GLUnurbs *nobj)                                                                                                                            |
| ti       | VOID           | gluEndPolygon(GLUtesselator *tess)                                                                                                                     |
| ti       | VOID           | gluEndSurface(GLUnurbs *nobj)                                                                                                                          |
| ti       | VOID           | gluEndTrim(GLUnurbs *nobj)                                                                                                                             |
| it       | CLubyte*       | gluErrorString(GLenum errCode)                                                                                                                         |
| titi     | VOID           | gluGetNurbsProperty(GLUnurbs *nobj, GLenum property, GLfloat *value)                                                                                   |
| it       | GLubyte*       | gluGetString(GLenum name)                                                                                                                              |
| titi     | VOID           | gluGetTessProperty(GLUtesselator *tess, GLenum which, GLdouble *value)                                                                                 |
| tfiii    | VOID           | gluLoadSamplingMatrices(GLUnurbs *nobj, const GLfloat modelMatrix[16], const GLfloat projMatrix[16], const GLint viewport[4])                          |
| dddddddi | VOID           | gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz) |
| t        | GLUnurbs*      | gluNewNurbsRenderer(void)                                                                                                                              |
| t        | GLUquadric*    | gluNewQuadric(void)                                                                                                                                    |
| t        | GLUtesselator* | gluNewTess(void)                                                                                                                                       |
| tii      | VOID           | gluNextContour(GLUtesselator *tess, GLenum type)                                                                                                       |

|              |      |                                                                                                                                                                                                               |
|--------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| titi         | VOID | <code>gluNurbsCallback</code> (GLUnurbs *nobj, GLenum which, void CALLBACK *fn)                                                                                                                               |
| tititiii     | VOID | <code>gluNurbsCurve</code> (GLUnurbs *nobj, GLint nknots, GLfloat *knot, GLint stride, GLfloat *ctarray, GLint order, GLenum type)                                                                            |
| tifi         | VOID | <code>gluNurbsProperty</code> (GLUnurbs *nobj, GLenum property, GLfloat value)                                                                                                                                |
| tititititiii | VOID | <code>gluNurbsSurface</code> (GLUnurbs *nobj, GLint sknot_count, float *sknot, GLint tknot_count, GLfloat *tknot, GLint s_stride, GLint t_stride, GLfloat *ctarray, GLint sorder, GLint torder, GLenum type)  |
| ddddi        | VOID | <code>gluOrtho2D</code> (GLdouble left, GLdouble right, GLdouble top, GLdouble bottom)                                                                                                                        |
| tddiiddi     | VOID | <code>gluPartialDisk</code> (GLUquadric *qobj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops, GLdouble startAngle, GLdouble sweepAngle)                                               |
| ddddi        | VOID | <code>gluPerspective</code> (GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar)                                                                                                                   |
| dddidi       | VOID | <code>gluPickMatrix</code> (GLdouble x, GLdouble y, GLdouble height, GLdouble width, GLint viewport[4])                                                                                                       |
| dddidditti   | int  | <code>gluProject</code> (GLdouble objx, GLdouble objy, GLdouble objz, const GLdouble modelMatrix[16], const GLdouble projMatrix[16], const GLint viewport[4], GLdouble *winx, GLdouble *winy, GLdouble *winz) |
| titiii       | VOID | <code>gluPwlCurve</code> (GLUnurbs *nobj, GLint count, GLfloat *array, GLint stride, GLenum type)                                                                                                             |
| titi         | VOID | <code>gluQuadricCallback</code> (GLUquadric *qobj, GLenum which, void CALLBACK *fn)                                                                                                                           |
| tii          | VOID | <code>gluQuadricDrawStyle</code> (GLUquadric *quadObject, GLenum drawStyle)                                                                                                                                   |
| tii          | VOID | <code>gluQuadricNormals</code> (GLUquadric *quadObject, GLenum normals)                                                                                                                                       |
| tii          | VOID | <code>gluQuadricOrientation</code> (GLUquadric *quadObject, GLenum orientation)                                                                                                                               |
| tii          | VOID | <code>gluQuadricTexture</code> (GLUquadric *quadObject, GLboolean textureCoords)                                                                                                                              |
| iiiiitiiti   | int  | <code>gluScaleMap</code> (GLenum format, GLint widthin, GLint heightin, GLenum typein, const void *datain, GLint widthout, GLint heightout, GLenum typeout, void *dataout)                                    |
| tdiii        | VOID | <code>gluSphere</code> (GLUquadric *qobj, GLdouble radius, GLint slices, GLint stacks)                                                                                                                        |
| ti           | VOID | <code>gluTessBeginContour</code> (GLUtesselator *tess)                                                                                                                                                        |
| tii          | VOID | <code>gluTessBeginPolygon</code> (GLUtesselator *tess, void *polygon_data)                                                                                                                                    |
| titi         | VOID | <code>gluTessCallback</code> (GLUtesselator *tess, GLenum which, void CALLBACK *fn)                                                                                                                           |
| ti           | VOID | <code>gluTessEndContour</code> (GLUtesselator *tess)                                                                                                                                                          |
|              |      |                                                                                                                                                                                                               |

|            |      |                                                                                                                                                                                                                 |
|------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti         | VOID | <code>gluTessEndPolygon</code> (GLUtesselator *tess)                                                                                                                                                            |
| tdddi      | VOID | <code>gluTessNormal</code> (GLUtesselator *tess, GLdouble x, GLdouble y, GLdouble z)                                                                                                                            |
| tidi       | VOID | <code>gluTessProperty</code> (GLUtesselator *tess, GLenum which, GLdouble value)                                                                                                                                |
| tdti       | VOID | <code>gluTessVertex</code> (GLUtesselator *tess, GLdouble coords[3], void *data)                                                                                                                                |
| dddddititi | int  | <code>gluUnProject</code> (GLdouble winx, GLdouble winy, GLdouble winz, const GLdouble modelMatrix[16], const GLdouble projMatrix[16], const GLint viewport[4], GLdouble *objx, GLdouble *objy, GLdouble *objz) |

# hid.dll

|          |         |                                                                                                                            |
|----------|---------|----------------------------------------------------------------------------------------------------------------------------|
| tuc      | BOOLEAN | HidD_FlushQueue(_In_ HANDLE HidDeviceObject)                                                                               |
| tuc      | BOOLEAN | HidD_FreePreparedData(_In_ PHIDP_PREPARED_DATA PreparedData)                                                               |
| ttuc     | BOOLEAN | HidD_GetAttributes(_In_ HANDLE HidDeviceObject,_Out_ PHIDD_ATTRIBUTES Attributes)                                          |
| ttuiuc   | BOOLEAN | HidD_GetConfiguration(_In_ HANDLE HidDeviceObject,_Out_ PHIDD_CONFIGURATION Configuration, _In_ ULONG ConfigurationLength) |
| ttuiuc   | BOOLEAN | HidD_GetFeature(_In_ HANDLE HidDeviceObject,_Out_ PVOID ReportBuffer,_In_ ULONG ReportBufferLength)                        |
| tuituiuc | BOOLEAN | HidD_GetIndexedString(_In_ HANDLE HidDeviceObject,_In_ ULONG StringIndex,_Out_ PVOID Buffer,_In_ ULONG BufferLength)       |
| ttuiuc   | BOOLEAN | HidD_GetInputReport(_In_ HANDLE HidDeviceObject,_Out_ PVOID ReportBuffer,_In_ ULONG ReportBufferLength)                    |
| ttuiuc   | BOOLEAN | HidD_GetManufacturerString(_In_ HANDLE HidDeviceObject,_Out_ PVOID Buffer,_In_ ULONG BufferLength)                         |
| ttuiuc   | BOOLEAN | HidD_GetMsGenreDescriptor(_In_ HANDLE HidDeviceObject,_Out_ PVOID Buffer,_In_ ULONG BufferLength)                          |
| ttuc     | BOOLEAN | HidD_GetNumInputBuffers(_In_ HANDLE HidDeviceObject,_Out_ PULONG NumberBuffers)                                            |
| ttuiuc   | BOOLEAN | HidD_GetPhysicalDescriptor(_In_ HANDLE HidDeviceObject,_Out_ PVOID Buffer,_In_ ULONG BufferLength)                         |
| ttuc     | BOOLEAN | HidD_GetPreparedData(_In_ HANDLE HidDeviceObject,_Out_ PHIDP_PREPARED_DATA * PreparedData)                                 |
| ttuiuc   | BOOLEAN | HidD_GetProductString(_In_ HANDLE HidDeviceObject,_Out_ PVOID Buffer,_In_ ULONG BufferLength)                              |
| ttuiuc   | BOOLEAN | HidD_GetSerialNumberString(_In_ HANDLE HidDeviceObject,_Out_ PVOID Buffer,_In_ ULONG BufferLength)                         |
| ttuiuc   | BOOLEAN | HidD_SetConfiguration(_In_ HANDLE HidDeviceObject,_In_ PHIDD_CONFIGURATION Configuration, _In_ ULONG ConfigurationLength)  |
| ttuiuc   | BOOLEAN | HidD_SetFeature(_In_ HANDLE HidDeviceObject,_In_ PVOID ReportBuffer,_In_ ULONG ReportBufferLength)                         |
| tuiuc    | BOOLEAN | HidD_SetNumInputBuffers(_In_ HANDLE HidDeviceObject,_In_ ULONG NumberBuffers)                                              |
| ttuiuc   | BOOLEAN | HidD_SetOutputReport(_In_ HANDLE HidDeviceObject,_In_ PVOID ReportBuffer,_In_ ULONG ReportBufferLength)                    |

|              |     |                                                                                                                                                                                                                                                            |
|--------------|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ittti        | int | HidP_GetButtonCaps(_In_ HIDP_REPORT_TYPE ReportType, _Out_ PHIDP_BUTTON_CAPS ButtonCaps, _Inout_ PUSHORT ButtonCapsLength, _In_ PHIDP_PREPARSED_DATA PreparedData)                                                                                         |
| tti          | int | HidP_GetCaps(_In_ PHIDP_PREPARSED_DATA PreparedData, _Out_ PHIDP_CAPS Capabilities)                                                                                                                                                                        |
| ittttuii     | int | HidP_GetData(_In_ HIDP_REPORT_TYPE ReportType, _Out_ PHIDP_DATA DataList, _Inout_ PULONG DataLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _Out_ PCHAR Report, _In_ ULONG ReportLength)                                                                  |
| iuhttti      | int | HidP_GetExtendedAttributes(_In_ HIDP_REPORT_TYPE ReportType, _In_ USHORT DataIndex, _In_ PHIDP_PREPARSED_DATA PreparedData, _Out_ PHIDP_EXTENDED_ATTRIBUTES Attributes, _Inout_ PULONG LengthAttributes)                                                   |
| ttti         | int | HidP_GetLinkCollectionNodes(_Out_ PHIDP_LINK_COLLECTION_NODE LinkCollectionNodes, _Inout_ PULONG LinkCollectionNodesLength, _In_ PHIDP_PREPARSED_DATA PreparedData)                                                                                        |
| iuuhuhtttuii | int | HidP_GetScaledUsageValue(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_ USAGE Usage, _Out_ PLONG UsageValue, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_ PCHAR Report, _In_ ULONG ReportLength)             |
| iuuhuhttti   | int | HidP_GetSpecificButtonCaps(_In_ HIDP_REPORT_TYPE ReportType, _In_opt_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_opt_ USAGE Usage, _Out_ PHIDP_BUTTON_CAPS ButtonCaps, _Inout_ PUSHORT ButtonCapsLength, _In_ PHIDP_PREPARSED_DATA PreparedData) |
| iuuhuhttti   | int | HidP_GetSpecificValueCaps(_In_ HIDP_REPORT_TYPE ReportType, _In_opt_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_opt_ USAGE Usage, _Out_ PHIDP_VALUE_CAPS ValueCaps, _Inout_ PUSHORT ValueCapsLength, _In_ PHIDP_PREPARSED_DATA PreparedData)     |
| iuuhttttuii  | int | HidP_GetUsages(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _Out_ PUSAGE UsageList, _Inout_ PULONG UsageLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _Out_ PCHAR Report, _In_ ULONG ReportLength)            |
| iuhttttuii   | int | HidP_GetUsagesEx(_In_ HIDP_REPORT_TYPE ReportType, _In_opt_ USHORT LinkCollection, _Inout_ PUSAGE_AND_PAGE ButtonList, _Inout_ ULONG * UsageLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_ PCHAR Report, _In_ ULONG ReportLength)                    |
| iuuhuhtttuii | int | HidP_GetUsageValue(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_ USAGE Usage, _Out_ PULONG UsageValue, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_ PCHAR Report, _In_ ULONG ReportLength)                  |

|                |       |                                                                                                                                                                                                                                                                                   |
|----------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| iuuhuhuhtttuii | int   | HidP_GetUsageValueArray(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_ USAGE Usage, _Inout_ PCHAR UsageValue, _In_ USHORT UsageValueByteLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_ PCHAR Report, _In_ ULONG ReportLength) |
| ittti          | int   | HidP_GetValueCaps(_In_ HIDP_REPORT_TYPE ReportType, _Out_ PHIDP_VALUE_CAPS ValueCaps, _Inout_ PUSHORT ValueCapsLength, _In_ PHIDP_PREPARSED_DATA PreparedData)                                                                                                                    |
| iucttuii       | int   | HidP_InitializeReportForID(_In_ HIDP_REPORT_TYPE ReportType, _In_ UCHAR ReportID, _In_ PHIDP_PREPARSED_DATA PreparedData, _Out_ PCHAR Report, _In_ ULONG ReportLength)                                                                                                            |
| itui           | ULONG | HidP_MaxDataListLength(_In_ HIDP_REPORT_TYPE ReportType, _In_ PHIDP_PREPARSED_DATA PreparedData)                                                                                                                                                                                  |
| iuhtui         | ULONG | HidP_MaxUsageListLength(_In_ HIDP_REPORT_TYPE ReportType, _In_opt_ USAGE UsagePage, _In_ PHIDP_PREPARSED_DATA PreparedData)                                                                                                                                                       |
| ittttuii       | int   | HidP_SetData(_In_ HIDP_REPORT_TYPE ReportType, _Inout_ PHIDP_DATA DataList, _Inout_ PULONG DataLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_reads_bytes_ReportLength PCHAR Report, _In_ ULONG ReportLength)                                                                |
| iuuhuhuittuii  | int   | HidP_SetScaledUsageValue(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_ USAGE Usage, _In_ LONG UsageValue, _In_ PHIDP_PREPARSED_DATA PreparedData, _Inout_ PCHAR Report, _In_ ULONG ReportLength)                                   |
| iuhuhtttuii    | int   | HidP_SetUsage(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _Inout_ PUSAGE UsageList, _Inout_ PULONG UsageLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_ PCHAR Report, _In_ ULONG ReportLength)                                   |
| iuuhuhuittuii  | int   | HidP_SetUsageValue(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_ USAGE Usage, _In_ ULONG UsageValue, _In_ PHIDP_PREPARSED_DATA PreparedData, _Inout_ PCHAR Report, _In_ ULONG ReportLength)                                        |
| iuuhuhuhtttuii | int   | HidP_SetUsageValueArray(_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _In_ USAGE Usage, _In_ PCHAR UsageValue, _In_ USHORT UsageValueByteLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _Inout_ PCHAR Report, _In_ ULONG ReportLength) |
| tuiitti        | int   | HidP_TranslateUsagesTo18042ScanCodes(_In_ PUSAGE ChangedUsageList, _In_ ULONG UsageListLength, _In_ HIDP_KEYBOARD_DIRECTION KeyAction, _Inout_ PHIDP_KEYBOARD_MODIFIER_STATE ModifierState, _In_ PHIDP_INSERT_SCANCODES InsertCodesProcedure, _In_opt_ PVOID InsertCodesContext)  |

|             |     |                                                                                                                                                                                                                                                                    |
|-------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| iuhhttttiii | int | <a href="#">HidP_UnsetUsage</a> (_In_ HIDP_REPORT_TYPE ReportType, _In_ USAGE UsagePage, _In_opt_ USHORT LinkCollection, _Inout_ PUSAGE UsageList, _Inout_ PULONG UsageLength, _In_ PHIDP_PREPARSED_DATA PreparedData, _In_ PCHAR Report, _In_ ULONG ReportLength) |
| ttttiii     | int | <a href="#">HidP_UsageListDifference</a> (_In_ PUSAGE PreviousUsageList, _In_ PUSAGE CurrentUsageList, _Out_ PUSAGE BreakUsageList, _Out_ PUSAGE MakeUsageList, _In_ ULONG UsageListLength)                                                                        |

## Kernel32.dll

|         |                      |                                            |
|---------|----------------------|--------------------------------------------|
| ttuiui  | long                 | _hread(HFILE hFile,LPVOID lpBuffer,long    |
| tauuiui | long                 | _hwrite(HFILE hFile,LPCSTR lpBuffer,long   |
| tt      | HFILE                | _lclose(HFILE hFile)                       |
| ait     | HFILE                | _lcreat(LPCSTR lpPathName,int iAttribute)  |
| tuiiui  | LONG                 | _lseek(HFILE hFile,LONG lOffset,int iOrig  |
| ait     | HFILE                | _lropen(LPCSTR lpPathName,int iReadWrite   |
| ttuiui  | UINT                 | _hread(HFILE hFile,LPVOID lpBuffer,UINT    |
| tauuiui | UINT                 | _hwrite(HFILE hFile,LPCSTR lpBuffer,UINT   |
| ti      | VOID                 | AcquireSRWLockExclusive(_Inout_ PSRWLOCK   |
| ti      | VOID                 | AcquireSRWLockShared(_Inout_ PSRWLOCK      |
| titi    | BOOL                 | ActivateActCtx(_In_ HANDLE hActCtx, _Out_  |
| suh     | ATOM                 | AddAtom(_In_ LPCTSTR lpString)             |
| auh     | ATOM                 | AddAtomA(_In_ LPCSTR lpString)             |
| wuh     | ATOM                 | AddAtomW(_In_ LPCWSTR lpString)            |
| sssi    | BOOL                 | AddConsoleAlias(_In_ LPCTSTR Source, _In_  |
| aaai    | BOOL                 | AddConsoleAliasA(_In_ LPCSTR Source, _In_  |
| wwwi    | BOOL                 | AddConsoleAliasW(_In_ LPCWSTR Source, _In_ |
| wt      | DLL_DIRECTORY_COOKIE | AddDllDirectory(_In_ PCWSTR NewDirectory   |
| titi    | BOOL                 | AddIntegrityLabelToBoundaryDescriptor(_In_ |
| ti      | VOID                 | AddRefActCtx(_In_ HANDLE hActCtx)          |
| ti      | BOOL                 | AddSecureMemoryCacheCallback(_In_ PSECURE  |
| titi    | BOOL                 | AddSIDToBoundaryDescriptor(_Inout_ HANDLE  |
| uitt    | PVOID                | AddVectoredContinueHandler(_In_ ULONG_PTR  |
| uitt    | PVOID                | AddVectoredExceptionHandler(_In_ ULONG_PTR |

|           |         |                                                                                                                                                                                           |
|-----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |         | VectoredHandler)                                                                                                                                                                          |
| tuiii     | BOOL    | AdjustCalendarDate(_Inout_ LPCALDATETIME lpCalDateTime, _In_ CALDATETIME_DATA _Out_ INT amount)                                                                                           |
| ttti      | BOOL    | AllocateUserPhysicalPages(_In_ HANDLE hProcess, _In_ PULONG_PTR NumberOfPages, _Out_ PULONG_PTR UserPfnArray)                                                                             |
| ttuii     | BOOL    | AllocateUserPhysicalPagesNuma(_In_ HANDLE hProcess, _Inout_ PULONG_PTR NumberOfPages, _Out_ PULONG_PTR PageArray, _In_ DWORD n...                                                         |
| i         | BOOL    | AllocConsole(void)                                                                                                                                                                        |
| ii        | VOID    | ApplicationRecoveryFinished(_In_ BOOL b...                                                                                                                                                |
| ti        | HRESULT | ApplicationRecoveryInProgress(_Out_ PBO...                                                                                                                                                |
| i         | BOOL    | AreFileApisANSI(void)                                                                                                                                                                     |
| i         | BOOL    | AreFileApisANSI(void)                                                                                                                                                                     |
| tti       | BOOL    | AssignProcessToJobObject(_In_ HANDLE hJob, HANDLE hProcess)                                                                                                                               |
| uii       | BOOL    | AttachConsole(_In_ DWORD dwProcessId)                                                                                                                                                     |
| ttuititi  | BOOL    | BackupRead(_In_ HANDLE hFile, _Out_ LPDWORD lpNumberOfBytesRead, _In_ BOOL bAbort, bProcessSecurity, _Out_ LPVOID *lpContext)                                                             |
| tuiuitti  | BOOL    | BackupSeek(_In_ HANDLE hFile, _In_ DWORD dwLowBytesToSeek, _In_ DWORD dwHighBytesToSeek, _Out_ LPDWORD lpdwLowByteSeeked, _Out_ LPDWORD lpdwHighByteSeeked, _In_ LPVOID *lpContext)       |
| ttuitiiti | BOOL    | BackupWrite(_In_ HANDLE hFile, _In_ LPVOID lpBuffer, _In_ DWORD nNumberOfBytesToWrite, _Out_ LPDWORD lpNumberOfBytesWritten, _In_ BOOL bAbort, bProcessSecurity, _Out_ LPVOID *lpContext) |
| uiiii     | BOOL    | Beep(_In_ DWORD dwFreq, _In_ DWORD dwDuration)                                                                                                                                            |
| sit       | HANDLE  | BeginUpdateResource(_In_ LPCTSTR pFileName, BOOL bDeleteExistingResources)                                                                                                                |
| ait       | HANDLE  | BeginUpdateResourceA(_In_ LPCSTR pFileName, BOOL bDeleteExistingResources)                                                                                                                |
| wit       | HANDLE  | BeginUpdateResourceW(_In_ LPCWSTR pFileName, BOOL bDeleteExistingResources)                                                                                                               |
| ttuii     | BOOL    | BindIoCompletionCallback(_In_ HANDLE hFile, LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine, _In_ ULONG Flags)                                                                        |
| sti       | BOOL    | BuildCommDCB(_In_ LPCTSTR lpDef, _Out_ LPDCB)                                                                                                                                             |

|             |      |                                                                                                                                                                                |
|-------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ati         | BOOL | BuildCommDCBA(_In_ LPCSTR lpDef, _Out_ LPDCB)                                                                                                                                  |
| stti        | BOOL | BuildCommDCBAndTimeouts(_In_ LPCTSTR LPDCB lpDCB, _Out_ LPCOMMTIMEOUTS lpCommTimeouts)                                                                                         |
| atti        | BOOL | BuildCommDCBAndTimeoutsA(_In_ LPCSTR LPDCB lpDCB, _Out_ LPCOMMTIMEOUTS lpCommTimeouts)                                                                                         |
| wtti        | BOOL | BuildCommDCBAndTimeoutsW(_In_ LPCWSTR LPDCB lpDCB, _Out_ LPCOMMTIMEOUTS lpCommTimeouts)                                                                                        |
| wti         | BOOL | BuildCommDCBW(_In_ LPCWSTR lpDef, _Out_ LPDCB)                                                                                                                                 |
| ti          | BOOL | CallbackMayRunLong(_Inout_ PTP_CALLBACK_INSTANCE pci)                                                                                                                          |
| stuituituii | BOOL | CallNamedPipe(_In_ LPCTSTR lpNamedPipe, LPVOID lpInBuffer, _In_ DWORD nInBufferSize, LPVOID lpOutBuffer, _In_ DWORD nOutBufferSize, LPDWORD lpBytesRead, _In_ DWORD nTimeout)  |
| atuituituii | BOOL | CallNamedPipeA(_In_ LPCSTR lpNamedPipe, LPVOID lpInBuffer, _In_ DWORD nInBufferSize, LPVOID lpOutBuffer, _In_ DWORD nOutBufferSize, LPDWORD lpBytesRead, _In_ DWORD nTimeout)  |
| wtuituituii | BOOL | CallNamedPipeW(_In_ LPCWSTR lpNamedPipe, LPVOID lpInBuffer, _In_ DWORD nInBufferSize, LPVOID lpOutBuffer, _In_ DWORD nOutBufferSize, LPDWORD lpBytesRead, _In_ DWORD nTimeout) |
| ti          | BOOL | CancelDeviceWakeupRequest(HANDLE hDevice)                                                                                                                                      |
| ti          | BOOL | Canceled(_In_ HANDLE hFile)                                                                                                                                                    |
| titi        | BOOL | CanceledEx(_In_ HANDLE hFile, _In_opt_ LPOVERLAPPED lpOverlapped)                                                                                                              |
| ti          | BOOL | CancelSynchronousIo(_In_ HANDLE hThread)                                                                                                                                       |
| ti          | VOID | CancelThreadpoolIo(_Inout_ PTP_IO pio)                                                                                                                                         |
| titi        | BOOL | CancelTimerQueueTimer(HANDLE TimerQueue, Timer)                                                                                                                                |
| ti          | BOOL | CancelWaitableTimer(_In_ HANDLE hTimer)                                                                                                                                        |
| ttuiuii     | BOOL | ChangeTimerQueueTimer(_In_opt_ HANDLE Timer, _Inout_ HANDLE Timer, _In_ ULONG DueTime, _In_ ULONG Period)                                                                      |
| sauitti     | BOOL | CheckNameLegalDOS8Dot(_In_ LPCTSTR lpName, _Out_opt_ LPSTR lpOemName, _In_ DWORD OemNameSize, _Out_opt_ PBOOL pbNameLegal, _Out_ PBOOL pbNameLegal)                            |

|           |         |                                                                                                                                                                                                   |
|-----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aauiitti  | BOOL    | CheckNameLegalDOS8Dot3A(_In_ LPCSTR lpName, _Out_opt_ LPSTR lpOemName, _In_ DWORD dwOemNameSize, _Out_opt_ PBOOL pbNameLegal, _Out_ PBOOL pbNameLegal)                                            |
| wauitti   | BOOL    | CheckNameLegalDOS8Dot3W(_In_ LPCWSTR lpName, _Out_opt_ LPSTR lpOemName, _In_ DWORD dwOemNameSize, _Out_opt_ PBOOL pbNameLegal, _Out_ PBOOL pbNameLegal)                                           |
| titi      | BOOL    | CheckRemoteDebuggerPresent(_In_ HANDLE hProcess, _Inout_ PBOOL pbDebuggerPresent)                                                                                                                 |
| ti        | BOOL    | ClearCommBreak(_In_ HANDLE hFile)                                                                                                                                                                 |
| titi      | BOOL    | ClearCommError(_In_ HANDLE hFile, _Out_ LPDWORD lpErrors, _Out_opt_ LPCOMMSTRTIMEOUT lpTimeouts)                                                                                                  |
| ti        | BOOL    | CloseHandle(_In_ HANDLE hObject)                                                                                                                                                                  |
| tuiuc     | BOOLEAN | ClosePrivateNamespace(_In_ HANDLE hProcess, _In_ ULONG Flags)                                                                                                                                     |
| ti        | VOID    | CloseThreadPool(_Inout_ PTP_POOL ptp)                                                                                                                                                             |
| ti        | VOID    | CloseThreadPoolCleanupGroup(_Inout_ PTP_CLEANUP_GROUP ptcg)                                                                                                                                       |
| titi      | VOID    | CloseThreadPoolCleanupGroupMembers(_In_ PTP_CLEANUP_GROUP ptcg, _In_ BOOL fCancelPendingCallbacks, _Inout_opt_ PVOID pvCleanupContext)                                                            |
| ti        | VOID    | CloseThreadpoolIo(_Inout_ PTP_IO pio)                                                                                                                                                             |
| ti        | VOID    | CloseThreadpoolTimer(_Inout_ PTP_TIMER ptp)                                                                                                                                                       |
| ti        | VOID    | CloseThreadpoolWait(_Inout_ PTP_WAIT ptp)                                                                                                                                                         |
| ti        | VOID    | CloseThreadpoolWork(_Inout_ PTP_WORK ptp)                                                                                                                                                         |
| stti      | BOOL    | CommConfigDialog(_In_ LPCTSTR lpszName, HWND hWnd, _Inout_ LPCOMMCONFIG lpCC)                                                                                                                     |
| atti      | BOOL    | CommConfigDialogA(_In_ LPCSTR lpszName, HWND hWnd, _Inout_ LPCOMMCONFIG lpCC)                                                                                                                     |
| wtti      | BOOL    | CommConfigDialogW(_In_ LPCWSTR lpszName, HWND hWnd, _Inout_ LPCOMMCONFIG lpCC)                                                                                                                    |
| ttui      | LONG    | CompareFileTime(_In_ const FILETIME *lpFileTime1, _In_ const FILETIME *lpFileTime2)                                                                                                               |
| uiuisisii | int     | CompareString(_In_ LCID Locale, _In_ DWORD dwCmpFlags, _In_ LPCTSTR lpString1, _In_ int cchCount1, _In_ LPCTSTR lpString2, _In_ int cchCount2)                                                    |
| uiuiaiaii | int     | CompareStringA(_In_ LCID Locale, _In_ DWORD dwCmpFlags, _In_ LPCSTR lpString1, _In_ int cchCount1, _In_ LPCSTR lpString2, _In_ int cchCount2)                                                     |
|           |         | CompareStringEx(_In_opt_ LPCWSTR lpLocaleName, _In_ DWORD dwCmpFlags, _In_ LPCTSTR lpString1, _In_ int cchCount1, _In_ LPCTSTR lpString2, _In_ int cchCount2, _In_ LCID Locale, _In_ int dwFlags) |

|             |        |                                                                                                                                                                                  |
|-------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wuiwiwittti | int    | DWORD dwCmpFlags, _In_ LPCWSTR lpSt<br>cchCount1, _In_ LPCWSTR lpString2, _In_<br>_In_opt_ LPNLSVERSIONINFO lpVersionI<br>_In_opt_ LPVOID lpReserved, _In_opt_ LP                |
| wiwiii      | int    | CompareStringOrdinal(_In_ LPCWSTR lpSt<br>cchCount1, _In_ LPCWSTR lpString2, _In_<br>_In_ BOOL bIgnoreCase)                                                                      |
| uiuiwiwii   | int    | CompareStringW(_In_ LCID Locale, _In_ D<br>dwCmpFlags, _In_ LPCWSTR lpString1, _In<br>cchCount1, _In_ LPCWSTR lpString2, _In_                                                    |
| tii         | BOOL   | ConnectNamedPipe(_In_ HANDLE hNamed<br>_Inout_opt_ LPOVERLAPPED lpOverlapp                                                                                                       |
| uiuiiii     | BOOL   | ContinueDebugEvent(_In_ DWORD dwProc<br>DWORD dwThreadId, _In_ DWORD dwCon                                                                                                       |
| tii         | BOOL   | ConvertCalDateTimeToSystemTime(_In_ co<br>LPCALDATETIME lpCalDateTime, _Out_ S<br>*lpSysTime)                                                                                    |
| uiui        | LCID   | ConvertDefaultLocale(_In_ LCID Locale)                                                                                                                                           |
| i           | BOOL   | ConvertFiberToThread(void)                                                                                                                                                       |
| tuiti       | BOOL   | ConvertSystemTimeToCalDateTime(_In_ co<br>SYSTEMTIME lpSysTime, _In_ CALID cal<br>LPCALDATETIME lpCalDateTime)                                                                   |
| tt          | LPVOID | ConvertThreadToFiber(_In_opt_ LPVOID lp                                                                                                                                          |
| tuit        | LPVOID | ConvertThreadToFiberEx(_In_opt_ LPVOID<br>_In_ DWORD dwFlags)                                                                                                                    |
| tuiti       | BOOL   | CopyContext(_Inout_ PCONTEXT Destinati<br>DWORD ContextFlags, _In_ PCONTEXT S                                                                                                    |
| ssii        | BOOL   | CopyFile(_In_ LPCTSTR lpExistingFileNam<br>LPCTSTR lpNewFileName, _In_ BOOL bFa                                                                                                  |
| aaai        | BOOL   | CopyFileA(_In_ LPCSTR lpExistingFileNam<br>LPCSTR lpNewFileName, _In_ BOOL bFail                                                                                                 |
| ssittuii    | BOOL   | CopyFileEx(_In_ LPCTSTR lpExistingFileN<br>LPCTSTR lpNewFileName, _In_opt_<br>LPPROGRESS_ROUTINE lpProgressRoutin<br>LPVOID lpData, _In_opt_ LPBOOL pbCanc<br>DWORD dwCopyFlags) |
| aatttiii    | BOOL   | CopyFileExA(_In_ LPCSTR lpExistingFileN<br>LPCSTR lpNewFileName, _In_opt_<br>LPPROGRESS_ROUTINE lpProgressRoutin<br>LPVOID lpData, _In_opt_ LPBOOL pbCanc<br>DWORD dwCopyFlags)  |
| wwtttiii    | BOOL   | CopyFileExW(_In_ LPCWSTR lpExistingFi<br>LPCWSTR lpNewFileName, _In_opt_<br>LPPROGRESS_ROUTINE lpProgressRoutin                                                                  |

|           |        |                                                                                                                                                                                                                       |
|-----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |        | LPVOID lpData, _In_opt_ LPBOOL pbCancel, DWORD dwCopyFlags)                                                                                                                                                           |
| sstttuiti | BOOL   | CopyFileTransacted(_In_ LPCTSTR lpExistingFileName, _In_ LPCTSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_opt_ LPBOOL pbCancel, DWORD dwCopyFlags, _In_ HANDLE hTransaction)  |
| aatttuiti | BOOL   | CopyFileTransactedA(_In_ LPCSTR lpExistingFileName, _In_ LPCSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_opt_ LPBOOL pbCancel, DWORD dwCopyFlags, _In_ HANDLE hTransaction)   |
| wwtttuiti | BOOL   | CopyFileTransactedW(_In_ LPCWSTR lpExistingFileName, _In_ LPCWSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_opt_ LPBOOL pbCancel, DWORD dwCopyFlags, _In_ HANDLE hTransaction) |
| wwii      | BOOL   | CopyFileW(_In_ LPCWSTR lpExistingFileName, _In_ LPCWSTR lpNewFileName, _In_ BOOL bFailIfExists)                                                                                                                       |
| iiui      | LONG   | CopyLZFile(INT,INT)                                                                                                                                                                                                   |
| tt        | HANDLE | CreateActCtx(_Inout_ PACTCTX pActCtx)                                                                                                                                                                                 |
| tt        | HANDLE | CreateActCtxA(_Inout_ PACTCTX pActCtx)                                                                                                                                                                                |
| tt        | HANDLE | CreateActCtxW(_Inout_ PACTCTX pActCtx)                                                                                                                                                                                |
| suit      | HANDLE | CreateBoundaryDescriptor(_In_ LPCTSTR lpName, ULONG Flags)                                                                                                                                                            |
| auit      | HANDLE | CreateBoundaryDescriptorA(_In_ LPCSTR lpName, ULONG Flags)                                                                                                                                                            |
| wuit      | HANDLE | CreateBoundaryDescriptorW(_In_ LPCWSTR lpName, ULONG Flags)                                                                                                                                                           |
| uiuituitt | HANDLE | CreateConsoleScreenBuffer(_In_ DWORD dwDesiredAccess, _In_ DWORD dwShareMode, const SECURITY_ATTRIBUTES *lpSecurityAttributes, DWORD dwFlags, _Reserved_ LPVOID lpScreenBufferData)                                   |
| sti       | BOOL   | CreateDirectory(_In_ LPCTSTR lpPathName, LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                                                                  |
| ati       | BOOL   | CreateDirectoryA(_In_ LPCSTR lpPathName, LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                                                                  |
| ssti      | BOOL   | CreateDirectoryEx(_In_ LPCTSTR lpTemplateName, _In_ LPCTSTR lpNewDirectory, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                      |
| aati      | BOOL   | CreateDirectoryExA(_In_ LPCSTR lpTemplateName, _In_ LPCSTR lpNewDirectory, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                       |

|             |        |                                                                                                                                                                                                                 |
|-------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwti        | BOOL   | CreateDirectoryExW(_In_ LPCWSTR lpTemplateName, _In_ LPCWSTR lpNewDirectory, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTemplateDirectory, _In_ DWORD dwFlags)                           |
| sstti       | BOOL   | CreateDirectoryTransacted(_In_opt_ LPCTSTR lpTemplateDirectory, _In_ LPCTSTR lpNewDirectory, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction, _In_ DWORD dwFlags)                 |
| aatti       | BOOL   | CreateDirectoryTransactedA(_In_opt_ LPCSTR lpTemplateDirectory, _In_ LPCSTR lpNewDirectory, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction, _In_ DWORD dwFlags)                  |
| wwtti       | BOOL   | CreateDirectoryTransactedW(_In_opt_ LPCWSTR lpTemplateDirectory, _In_ LPCWSTR lpNewDirectory, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction, _In_ DWORD dwFlags)                |
| wti         | BOOL   | CreateDirectoryW(_In_ LPCWSTR lpPathName, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTemplateDirectory, _In_ DWORD dwFlags)                                                              |
| tiist       | HANDLE | CreateEvent(_In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes, _In_ BOOL bManualReset, _In_ BOOL bInitialState, _In_opt_ LPCTSTR lpName)                                                                         |
| tiiat       | HANDLE | CreateEventA(_In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes, _In_ BOOL bManualReset, _In_ BOOL bInitialState, _In_opt_ LPCSTR lpName)                                                                         |
| tsuiuit     | HANDLE | CreateEventEx(_In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes, _In_opt_ LPCTSTR lpName, _In_ DWORD dwFlags, _In_ DWORD dwDesiredAccess)                                                                        |
| tauuiit     | HANDLE | CreateEventExA(_In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes, _In_opt_ LPCSTR lpName, _In_ DWORD dwFlags, _In_ DWORD dwDesiredAccess)                                                                        |
| twuiuit     | HANDLE | CreateEventExW(_In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes, _In_opt_ LPCWSTR lpName, _In_ DWORD dwFlags, _In_ DWORD dwDesiredAccess)                                                                       |
| tiiwt       | HANDLE | CreateEventW(_In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes, _In_ BOOL bManualReset, _In_ BOOL bInitialState, _In_opt_ LPCWSTR lpName)                                                                        |
| ttt         | LPVOID | CreateFiber(_In_ SIZE_T dwStackSize, _In_ LPFIBER_START_ROUTINE lpStartAddress, LPVOID lpParameter)                                                                                                             |
| ttuitt      | LPVOID | CreateFiberEx(_In_ SIZE_T dwStackCommitment, _In_ SIZE_T dwStackReserveSize, _In_ DWORD dwFlags, _In_ LPFIBER_START_ROUTINE lpStartAddress, LPVOID lpParameter)                                                 |
| suiuituiuit | HANDLE | CreateFile(_In_ LPCTSTR lpFileName, _In_ DWORD dwDesiredAccess, _In_ DWORD dwShareMode, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD dwCreationDisposition, _In_ DWORD dwFlagsAndAttributes) |

|                |        |                                                                                                                                                                                                                                                                                                 |
|----------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                |        | dwFlagsAndAttributes, _In_opt_ HANDLE l                                                                                                                                                                                                                                                         |
| auiuituiuitt   | HANDLE | CreateFileA(_In_ LPCSTR lpFileName, _In_ dwDesiredAccess, _In_ DWORD dwShareM LPSECURITY_ATTRIBUTES lpSecurityAt DWORD dwCreationDisposition, _In_ DW dwFlagsAndAttributes, _In_opt_ HANDLE l                                                                                                   |
| ttuiuiuist     | HANDLE | CreateFileMapping(_In_ HANDLE hFile, _ LPSECURITY_ATTRIBUTES lpAttributes, flProtect, _In_ DWORD dwMaximumSizeHi DWORD dwMaximumSizeLow, _In_opt_ L lpName)                                                                                                                                     |
| ttuiuiuiait    | HANDLE | CreateFileMappingA(_In_ HANDLE hFile, _ LPSECURITY_ATTRIBUTES lpAttributes, flProtect, _In_ DWORD dwMaximumSizeHi DWORD dwMaximumSizeLow, _In_opt_ L                                                                                                                                            |
| ttuiuiuisuit   | HANDLE | CreateFileMappingNuma(_In_ HANDLE hF LPSECURITY_ATTRIBUTES lpFileMappir _In_ DWORD flProtect, _In_ DWORD dwMaximumSizeHigh, _In_ DWORD dwMa _In_opt_ LPCTSTR lpName, _In_ DWORD                                                                                                                 |
| ttuiuiuiauit   | HANDLE | CreateFileMappingNumaA(_In_ HANDLE h LPSECURITY_ATTRIBUTES lpFileMappir _In_ DWORD flProtect, _In_ DWORD dwMaximumSizeHigh, _In_ DWORD dwMa _In_opt_ LPCSTR lpName, _In_ DWORD n                                                                                                                |
| ttuiuiuiwuit   | HANDLE | CreateFileMappingNumaW(_In_ HANDLE LPSECURITY_ATTRIBUTES lpFileMappir _In_ DWORD flProtect, _In_ DWORD dwMaximumSizeHigh, _In_ DWORD dwMa _In_opt_ LPCWSTR lpName, _In_ DWORD                                                                                                                   |
| ttuiuiuigt     | HANDLE | CreateFileMappingW(_In_ HANDLE hFile, LPSECURITY_ATTRIBUTES lpAttributes, flProtect, _In_ DWORD dwMaximumSizeHi DWORD dwMaximumSizeLow, _In_opt_ L lpName)                                                                                                                                      |
| suiuituiuitttt | HANDLE | CreateFileTransacted(_In_ LPCTSTR lpFileI DWORD dwDesiredAccess, _In_ DWORD d _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD dwCreat _In_ DWORD dwFlagsAndAttributes, _In_o hTemplateFile, _In_ HANDLE hTransaction, PUSHORT pusMiniVersion, _Reserved_ PV pExtendedParameter) |
| auiuituiuitttt | HANDLE | CreateFileTransactedA(_In_ LPCSTR lpFileI DWORD dwDesiredAccess, _In_ DWORD d _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD dwCreat _In_ DWORD dwFlagsAndAttributes, _In_o                                                                                                    |

|                |        |                                                                                                                                                                                                                                                                                                                                                |
|----------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                |        | hTemplateFile, _In_ HANDLE hTransaction, PUSHORT pusMiniVersion, _Reserved_ PV(pExtendedParameter)                                                                                                                                                                                                                                             |
| wuiuituiuitttt | HANDLE | CreateFileTransactedW(_In_ LPCWSTR lpFileName, _In_ DWORD dwDesiredAccess, _In_ DWORD dwShareMode, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD dwCreationDisposition, _In_ DWORD dwFlagsAndAttributes, _In_opt_ HANDLE hTemplateFile, _In_ HANDLE hTransaction, PUSHORT pusMiniVersion, _Reserved_ PV(pExtendedParameter)) |
| wuiuituiuitt   | HANDLE | CreateFileW(_In_ LPCWSTR lpFileName, _In_ DWORD dwDesiredAccess, _In_ DWORD dwShareMode, LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD dwCreationDisposition, _In_ DWORD dwFlagsAndAttributes, _In_opt_ HANDLE hTemplateFile, _In_ HANDLE hTransaction, PUSHORT pusMiniVersion, _Reserved_ PV(pExtendedParameter))                    |
| ssti           | BOOL   | CreateHardLink(_In_ LPCTSTR lpFileName, _In_ LPCTSTR lpExistingFileName, _Reserved_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction)                                                                                                                                                                                      |
| aati           | BOOL   | CreateHardLinkA(_In_ LPCSTR lpFileName, _In_ LPCSTR lpExistingFileName, _Reserved_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction)                                                                                                                                                                                       |
| ssti           | BOOL   | CreateHardLinkTransacted(_In_ LPCTSTR lpFileName, _In_ LPCTSTR lpExistingFileName, _Reserved_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction)                                                                                                                                                                            |
| aati           | BOOL   | CreateHardLinkTransactedA(_In_ LPCSTR lpFileName, _In_ LPCSTR lpExistingFileName, _Reserved_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction)                                                                                                                                                                             |
| wwtti          | BOOL   | CreateHardLinkTransactedW(_In_ LPCWSTR lpFileName, _In_ LPCWSTR lpExistingFileName, _Reserved_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction)                                                                                                                                                                           |
| wwti           | BOOL   | CreateHardLinkW(_In_ LPCWSTR lpFileName, _In_ LPCWSTR lpExistingFileName, _Reserved_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ HANDLE hTransaction)                                                                                                                                                                                     |
| tttuit         | HANDLE | CreateIoCompletionPort(_In_ HANDLE FileHandle, _In_opt_ HANDLE ExistingCompletionPort, ULONG_PTR CompletionKey, _In_ DWORD NumberOfConcurrentThreads)                                                                                                                                                                                          |
| tst            | HANDLE | CreateJobObject(_In_opt_ LPSECURITY_ATTRIBUTES lpJobAttributes, _In_opt_ LPCTSTR lpName)                                                                                                                                                                                                                                                       |
| tat            | HANDLE | CreateJobObjectA(_In_opt_ LPSECURITY_ATTRIBUTES lpJobAttributes, _In_opt_ LPCSTR lpName)                                                                                                                                                                                                                                                       |
| twt            | HANDLE | CreateJobObjectW(_In_opt_ LPSECURITY_ATTRIBUTES lpJobAttributes, _In_opt_ LPCWSTR lpName)                                                                                                                                                                                                                                                      |

|                  |        |                                                                                                                                                                                                                                            |
|------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  |        | lpJobAttributes, _In_opt_ LPCWSTR lpName)                                                                                                                                                                                                  |
| uituii           | BOOL   | CreateJobSet(ULONG NumJob, PJOB_SET UserJobSet, ULONG Flags)                                                                                                                                                                               |
| suiuiitt         | HANDLE | CreateMailslot(_In_ LPCTSTR lpName, _In_ nMaxMessageSize, _In_ DWORD lReadTime, LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                                                |
| auuiiitt         | HANDLE | CreateMailslotA(_In_ LPCSTR lpName, _In_ nMaxMessageSize, _In_ DWORD lReadTime, LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                                                |
| wuiuiitt         | HANDLE | CreateMailslotW(_In_ LPCWSTR lpName, _In_ nMaxMessageSize, _In_ DWORD lReadTime, LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                                               |
| uit              | HANDLE | CreateMemoryResourceNotification(_In_ MEMORY_RESOURCE_NOTIFICATION_TYPE NotificationType)                                                                                                                                                  |
| tist             | HANDLE | CreateMutex(_In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes, _In_ BOOL bInitialOwner, LPCTSTR lpName)                                                                                                                                     |
| tiait            | HANDLE | CreateMutexA(_In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes, _In_ BOOL bInitialOwner, LPCSTR lpName)                                                                                                                                     |
| tsuiiitt         | HANDLE | CreateMutexEx(_In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes, _In_opt_ LPCTSTR lpName, DWORD dwFlags, _In_ DWORD dwDesiredAccess)                                                                                                        |
| tauuiitt         | HANDLE | CreateMutexExA(_In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes, _In_opt_ LPCSTR lpName, DWORD dwFlags, _In_ DWORD dwDesiredAccess)                                                                                                        |
| twuiiitt         | HANDLE | CreateMutexExW(_In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes, _In_opt_ LPCWSTR lpName, DWORD dwFlags, _In_ DWORD dwDesiredAccess)                                                                                                       |
| tiwt             | HANDLE | CreateMutexW(_In_opt_ LPSECURITY_ATTRIBUTES lpMutexAttributes, _In_ BOOL bInitialOwner, LPCWSTR lpName)                                                                                                                                    |
| suiuiuiuiuiiitt  | HANDLE | CreateNamedPipe(_In_ LPCTSTR lpName, _In_ dwOpenMode, _In_ DWORD dwPipeMode, _In_ nMaxInstances, _In_ DWORD nOutBufferSize, _In_ DWORD nInBufferSize, _In_ DWORD nDefaultQueueLength, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes) |
| auuiuiuiuiuiiitt | HANDLE | CreateNamedPipeA(_In_ LPCSTR lpName, _In_ dwOpenMode, _In_ DWORD dwPipeMode, _In_ nMaxInstances, _In_ DWORD nOutBufferSize, _In_ DWORD nInBufferSize, _In_ DWORD nDefaultQueueLength, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes) |
|                  |        |                                                                                                                                                                                                                                            |

|                |        |                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wuiuiuiuiuiitt | HANDLE | CreateNamedPipeW(_In_ LPCWSTR lpName, _In_ DWORD dwOpenMode, _In_ DWORD dwProcessId, _In_ DWORD nMaxInstances, _In_ DWORD nOpenTimeout, _In_ DWORD nInBufferSize, _In_ DWORD nOutBufferSize, _In_ DWORD nDefaultTimeOut, _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes)                                                                                                                 |
| tttiii         | BOOL   | CreatePipe(_Out_ PHANDLE hReadPipe, _Out_ PHANDLE hWritePipe, _In_opt_ LPSECURITY_ATTRIBUTES lpPipeAttributes, _In_ DWORD nSize)                                                                                                                                                                                                                                                              |
| ttst           | HANDLE | CreatePrivateNamespace(_In_opt_ LPSECURITY_ATTRIBUTES lpPrivateNamespaceAttributes, _In_ LPVOID lpBoundaryDescriptor, _In_ LPCTSTR lpAlias)                                                                                                                                                                                                                                                   |
| ttat           | HANDLE | CreatePrivateNamespaceA(_In_opt_ LPSECURITY_ATTRIBUTES lpPrivateNamespaceAttributes, _In_ LPVOID lpBoundaryDescriptor, _In_ LPCSTR lpAlias)                                                                                                                                                                                                                                                   |
| ttwt           | HANDLE | CreatePrivateNamespaceW(_In_opt_ LPSECURITY_ATTRIBUTES lpPrivateNamespaceAttributes, _In_ LPVOID lpBoundaryDescriptor, _In_ LPCWSTR lpAlias)                                                                                                                                                                                                                                                  |
| ssttiuitsti    | BOOL   | CreateProcess(_In_opt_ LPCTSTR lpApplicationName, _Inout_opt_ LPTSTR lpCommandLine, _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, _In_ DWORD dwCreationFlags, _In_opt_ LPVOID lpEnvironment, _In_opt_ lpCurrentDirectory, _In_ LPSTARTUPINFO lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation)  |
| aattiiutatti   | BOOL   | CreateProcessA(_In_opt_ LPCSTR lpApplicationName, _Inout_opt_ LPSTR lpCommandLine, _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, _In_ DWORD dwCreationFlags, _In_opt_ LPVOID lpEnvironment, _In_opt_ lpCurrentDirectory, _In_ LPSTARTUPINFO lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation)   |
| wwtiiiutwti    | BOOL   | CreateProcessW(_In_opt_ LPCWSTR lpApplicationName, _Inout_opt_ LPWSTR lpCommandLine, _In_opt_ LPSECURITY_ATTRIBUTES lpProcessAttributes, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, _In_ DWORD dwCreationFlags, _In_opt_ LPVOID lpEnvironment, _In_opt_ lpCurrentDirectory, _In_ LPSTARTUPINFO lpStartupInfo, _Out_ LPPROCESS_INFORMATION lpProcessInformation) |
| ttttuitt       | HANDLE | CreateRemoteThread(_In_ HANDLE hProcess, _In_opt_ LPSECURITY_ATTRIBUTES lpThreadAttributes, _In_ SIZE_T dwStackSize, _In_ LPTHREAD_START_ROUTINE lpThreadFunction, _In_opt_ LPVOID lpParameter, _In_ DWORD dwFlags)                                                                                                                                                                           |

|             |         |                                                                                                                                                                                                                                                                        |
|-------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |         | lpStartAddress, _In_ LPVOID lpParameter, _dwCreationFlags, _Out_ LPDWORD lpThre                                                                                                                                                                                        |
| ttttuittt   | HANDLE  | CreateRemoteThreadEx(_In_ HANDLE hPro<br>LPSECURITY_ATTRIBUTES lpThreadAttr<br>SIZE_T dwStackSize, _In_ LPTHREAD_ST<br>lpStartAddress, _In_opt_ LPVOID lpParame<br>DWORD dwCreationFlags, _In_opt_<br>LPPROC_THREAD_ATTRIBUTE_LIST lp<br>_Out_opt_ LPDWORD lpThreadId) |
| tuiuist     | HANDLE  | CreateSemaphore(_In_opt_ LPSECURITY_<br>lpSemaphoreAttributes, _In_ LONG lInitialC<br>LONG lMaximumCount, _In_opt_ LPCTST                                                                                                                                              |
| tuiuiat     | HANDLE  | CreateSemaphoreA(_In_opt_ LPSECURITY_<br>lpSemaphoreAttributes, _In_ LONG lInitialC<br>LONG lMaximumCount, _In_opt_ LPCSTR                                                                                                                                             |
| tuiuisuiuit | HANDLE  | CreateSemaphoreEx(_In_opt_<br>LPSECURITY_ATTRIBUTES lpSemaphore<br>LONG lInitialCount, _In_ LONG lMaximum<br>LPCTSTR lpName, _Reserved_ DWORD dw<br>DWORD dwDesiredAccess)                                                                                             |
| tuiuiauiuit | HANDLE  | CreateSemaphoreExA(_In_opt_<br>LPSECURITY_ATTRIBUTES lpSemaphore<br>LONG lInitialCount, _In_ LONG lMaximum<br>LPCSTR lpName, _Reserved_ DWORD dwl<br>DWORD dwDesiredAccess)                                                                                            |
| tuiuiwuiuit | HANDLE  | CreateSemaphoreExW(_In_opt_<br>LPSECURITY_ATTRIBUTES lpSemaphore<br>LONG lInitialCount, _In_ LONG lMaximum<br>LPCWSTR lpName, _Reserved_ DWORD d<br>DWORD dwDesiredAccess)                                                                                             |
| tuiuiwt     | HANDLE  | CreateSemaphoreW(_In_opt_ LPSECURITY_<br>lpSemaphoreAttributes, _In_ LONG lInitialC<br>LONG lMaximumCount, _In_opt_ LPCWST                                                                                                                                             |
| ssuiuc      | BOOLEAN | CreateSymbolicLink(_In_ LPTSTR lpSymli<br>_In_ LPTSTR lpTargetFileName, _In_ DWC                                                                                                                                                                                       |
| aauiuc      | BOOLEAN | CreateSymbolicLinkA(_In_ LPSTR lpSymli<br>_In_ LPSTR lpTargetFileName, _In_ DWOR                                                                                                                                                                                       |
| ssuituc     | BOOLEAN | CreateSymbolicLinkTransacted(_In_ LPTST<br>lpSymlinkFileName, _In_ LPTSTR lpTarget<br>DWORD dwFlags, _In_ HANDLE hTransac                                                                                                                                              |
| aauituc     | BOOLEAN | CreateSymbolicLinkTransactedA(_In_ LPST<br>lpSymlinkFileName, _In_ LPSTR lpTargetFi<br>DWORD dwFlags, _In_ HANDLE hTransac                                                                                                                                             |
| wwuituc     | BOOLEAN | CreateSymbolicLinkTransactedW(_In_ LPW<br>lpSymlinkFileName, _In_ LPWSTR lpTarget<br>DWORD dwFlags, _In_ HANDLE hTransac                                                                                                                                               |

|             |                   |                                                                                                                                                                                                 |
|-------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwuiuc      | BOOLEAN           | CreateSymbolicLinkW(_In_ LPWSTR lpSym<br>_In_ LPWSTR lpTargetFileName, _In_ DWO                                                                                                                 |
| tuiuiuiui   | DWORD             | CreateTapePartition(_In_ HANDLE hDevice<br>dwPartitionMethod, _In_ DWORD dwCount<br>dwSize)                                                                                                     |
| ttttuitt    | HANDLE            | CreateThread(_In_opt_ LPSECURITY_ATT<br>lpThreadAttributes, _In_ SIZE_T dwStackSi<br>LPTHREAD_START_ROUTINE lpStartAdd<br>LPVOID lpParameter, _In_ DWORD dwCre<br>_Out_opt_ LPDWORD lpThreadId) |
| tt          | PTP_POOL          | CreateThreadpool(_Reserved_ PVOID reserv                                                                                                                                                        |
| t           | PTP_CLEANUP_GROUP | CreateThreadpoolCleanupGroup(void)                                                                                                                                                              |
| tttt        | PTP_IO            | CreateThreadpoolIo(_In_ HANDLE fl, _In_<br>PTP_WIN32_IO_CALLBACK pfnio, _Inout<br>pv, _In_opt_ PTP_CALLBACK_ENVIRON                                                                             |
| tttt        | PTP_TIMER         | CreateThreadpoolTimer(_In_ PTP_TIMER_<br>pfnTi, _Inout_opt_ PVOID pv, _In_opt_<br>PTP_CALLBACK_ENVIRON pcbe)                                                                                    |
| tttt        | PTP_WAIT          | CreateThreadpoolWait(_In_ PTP_WAIT_CA<br>pfnwa, _Inout_opt_ PVOID pv, _In_opt_<br>PTP_CALLBACK_ENVIRON pcbe)                                                                                    |
| tttt        | PTP_WORK          | CreateThreadpoolWork(_In_ PTP_WORK_C<br>pfnwk, _Inout_opt_ PVOID pv, _In_opt_<br>PTP_CALLBACK_ENVIRON pcbe)                                                                                     |
| t           | HANDLE            | CreateTimerQueue(void)                                                                                                                                                                          |
| ttttuiuiuii | BOOL              | CreateTimerQueueTimer(_Out_ PHANDLE<br>_In_opt_ HANDLE TimerQueue, _In_<br>WAITORTIMERCALLBACK Callback, _In_<br>Parameter, _In_ DWORD DueTime, _In_ DV<br>_In_ ULONG Flags)                    |
| uiuit       | HANDLE            | CreateToolhelp32Snapshot(_In_ DWORD dv<br>DWORD th32ProcessID)                                                                                                                                  |
| tist        | HANDLE            | CreateWaitableTimer(_In_opt_<br>LPSECURITY_ATTRIBUTES lpTimerAttri<br>BOOL bManualReset, _In_opt_ LPCTSTR lp                                                                                    |
| tiait       | HANDLE            | CreateWaitableTimerA(_In_opt_<br>LPSECURITY_ATTRIBUTES lpTimerAttri<br>BOOL bManualReset, _In_opt_ LPCSTR lpT                                                                                   |
| tsuiuit     | HANDLE            | CreateWaitableTimerEx(_In_opt_<br>LPSECURITY_ATTRIBUTES lpTimerAttri<br>LPCTSTR lpTimerName, _In_ DWORD dw<br>DWORD dwDesiredAccess)                                                            |
| tauuiuit    | HANDLE            | CreateWaitableTimerExA(_In_opt_<br>LPSECURITY_ATTRIBUTES lpTimerAttri<br>LPCSTR lpTimerName, _In_ DWORD dwFl                                                                                    |

|         |        |                                                                                                                                        |
|---------|--------|----------------------------------------------------------------------------------------------------------------------------------------|
|         |        | DWORD dwDesiredAccess)                                                                                                                 |
| twuiuit | HANDLE | CreateWaitableTimerExW(_In_opt_<br>LPSECURITY_ATTRIBUTES lpTimerAttrib<br>LPCWSTR lpTimerName, _In_ DWORD dw<br>DWORD dwDesiredAccess) |
| tiwt    | HANDLE | CreateWaitableTimerW(_In_opt_<br>LPSECURITY_ATTRIBUTES lpTimerAttrib<br>BOOL bManualReset, _In_opt_ LPCWSTR l                          |
| uiti    | BOOL   | DeactivateActCtx(_In_ DWORD dwFlags, _<br>ULONG_PTR ulCookie)                                                                          |
| uii     | BOOL   | DebugActiveProcess(_In_ DWORD dwProce                                                                                                  |
| uii     | BOOL   | DebugActiveProcessStop(_In_ DWORD dw                                                                                                   |
| i       | VOID   | DebugBreak(void)                                                                                                                       |
| ti      | BOOL   | DebugBreakProcess(_In_ HANDLE Process)                                                                                                 |
| ii      | BOOL   | DebugSetProcessKillOnExit(_In_ BOOL Kill                                                                                               |
| uissi   | BOOL   | DefineDosDevice(_In_ DWORD dwFlags, _<br>lpDeviceName, _In_opt_ LPCTSTR lpTarget                                                       |
| uiaai   | BOOL   | DefineDosDeviceA(_In_ DWORD dwFlags,<br>lpDeviceName, _In_opt_ LPCSTR lpTargetP                                                        |
| uiwwi   | BOOL   | DefineDosDeviceW(_In_ DWORD dwFlags,<br>LPCWSTR lpDeviceName, _In_opt_ LPCW<br>lpTargetPath)                                           |
| uhuh    | ATOM   | DeleteAtom(_In_ ATOM nAtom)                                                                                                            |
| ti      | VOID   | DeleteBoundaryDescriptor(_In_ HANDLE<br>BoundaryDescriptor)                                                                            |
| ti      | VOID   | DeleteCriticalSection(_Inout_ LPCRITICAL<br>lpCriticalSection)                                                                         |
| ti      | VOID   | DeleteFiber(_In_ LPVOID lpFiber)                                                                                                       |
| si      | BOOL   | DeleteFile(_In_ LPCTSTR lpFileName)                                                                                                    |
| ai      | BOOL   | DeleteFileA(_In_ LPCSTR lpFileName)                                                                                                    |
| sti     | BOOL   | DeleteFileTransacted(_In_ LPCTSTR lpFile<br>HANDLE hTransaction)                                                                       |
| ati     | BOOL   | DeleteFileTransactedA(_In_ LPCSTR lpFile<br>HANDLE hTransaction)                                                                       |
| wti     | BOOL   | DeleteFileTransactedW(_In_ LPCWSTR lpF<br>HANDLE hTransaction)                                                                         |
| wi      | BOOL   | DeleteFileW(_In_ LPCWSTR lpFileName)                                                                                                   |
| ti      | VOID   | DeleteProcThreadAttributeList(_Inout_<br>LPPROC_THREAD_ATTRIBUTE_LIST lp                                                               |
| ti      | BOOL   | DeleteTimerQueue(_In_ HANDLE TimerQu                                                                                                   |



|           |      |                                                                                                                                                                                     |
|-----------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii       | BOOL | EndUpdateResourceW(_In_ HANDLE hUpdateResource, _In_ DWORD dwDiscard)                                                                                                               |
| ti        | VOID | EnterCriticalSection(_Inout_ LPCRITICAL_SECTION lpCriticalSection)                                                                                                                  |
| tuiuiiii  | BOOL | EnumCalendarInfo(_In_ CALINFO_ENUMPROC pCalInfoEnumProc, _In_ LCID Locale, _In_ Calendar, _In_ CALTYPE CalType)                                                                     |
| tuiuiiii  | BOOL | EnumCalendarInfoA(_In_ CALINFO_ENUMPROC pCalInfoEnumProc, _In_ LCID Locale, _In_ Calendar, _In_ CALTYPE CalType)                                                                    |
| tuiuiiii  | BOOL | EnumCalendarInfoEx(_In_ CALINFO_ENUMPROC pCalInfoEnumProcEx, _In_ LCID Locale, _In_ Calendar, _In_ CALTYPE CalType)                                                                 |
| tuiuiiii  | BOOL | EnumCalendarInfoExA(_In_ CALINFO_ENUMPROC pCalInfoEnumProcEx, _In_ LCID Locale, _In_ Calendar, _In_ CALTYPE CalType)                                                                |
| twuiwuiti | BOOL | EnumCalendarInfoExEx(_In_ CALINFO_ENUMPROC pCalInfoEnumProcEx, _In_opt_ LPCWSTR lpLocaleName, _In_ Calendar, _In_opt_ LPCWSTR lpReserved, _In_ CALTYPE CalType, _In_ LPARAM lParam) |
| tuiuiiii  | BOOL | EnumCalendarInfoExW(_In_ CALINFO_ENUMPROC pCalInfoEnumProcEx, _In_ LCID Locale, _In_ Calendar, _In_ CALTYPE CalType)                                                                |
| tuiuiiii  | BOOL | EnumCalendarInfoW(_In_ CALINFO_ENUMPROC pCalInfoEnumProc, _In_ LCID Locale, _In_ Calendar, _In_ CALTYPE CalType)                                                                    |
| tuiuiii   | BOOL | EnumDateFormats(_In_ DATEFMT_ENUMPROC lpDateFmtEnumProc, _In_ LCID Locale, _In_ dwFlags)                                                                                            |
| tuiuiii   | BOOL | EnumDateFormatsA(_In_ DATEFMT_ENUMPROC lpDateFmtEnumProc, _In_ LCID Locale, _In_ dwFlags)                                                                                           |
| tuiuiii   | BOOL | EnumDateFormatsEx(_In_ DATEFMT_ENUMPROC lpDateFmtEnumProcEx, _In_ LCID Locale, _In_ dwFlags)                                                                                        |
| tuiuiii   | BOOL | EnumDateFormatsExA(_In_ DATEFMT_ENUMPROC lpDateFmtEnumProcEx, _In_ LCID Locale, _In_ dwFlags)                                                                                       |
| twuiti    | BOOL | EnumDateFormatsExEx(_In_ DATEFMT_ENUMPROC pDateFmtEnumProcEx, _In_opt_ LPCWSTR lpLocaleName, _In_ DATEFMT_ENUMPROC pDateFmtEnumProc, _In_ dwFlags, _In_ LPARAM lParam)              |
| tuiuiii   | BOOL | EnumDateFormatsExW(_In_ DATEFMT_ENUMPROC lpDateFmtEnumProcEx, _In_ LCID Locale, _In_ dwFlags)                                                                                       |

|            |      |                                                                                                                                                           |
|------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |      | dwFlags)                                                                                                                                                  |
| tuiiii     | BOOL | EnumDateFormatsW(_In_ DATEFMT_ENUMPROC lpDateFmtEnumProc, _In_ LCID Locale, _In_ dwFlags)                                                                 |
| tuiuiti    | BOOL | EnumLanguageGroupLocales(_In_ LANGGROUPLOCALE_ENUMPROC lpLangGroupLocaleEnumProc, _In_ LGRPID LanguageGroup, _In_ DWORD dwFlags, _In_ IParam)             |
| tuiuiti    | BOOL | EnumLanguageGroupLocalesA(_In_ LANGGROUPLOCALE_ENUMPROC lpLangGroupLocaleEnumProc, _In_ LGRPID LanguageGroup, _In_ DWORD dwFlags, _In_ IParam)            |
| tuiuiti    | BOOL | EnumLanguageGroupLocalesW(_In_ LANGGROUPLOCALE_ENUMPROC lpLangGroupLocaleEnumProc, _In_ LGRPID LanguageGroup, _In_ DWORD dwFlags, _In_ IParam)            |
| tsstti     | BOOL | EnumResourceLanguages(_In_ HMODULE LPCTSTR lpType, _In_ LPCTSTR lpName, ENUMRESLANGPROC lpEnumFunc, _In_ IParam)                                          |
| taatti     | BOOL | EnumResourceLanguagesA(_In_ HMODULE LPCSTR lpType, _In_ LPCSTR lpName, _In_ ENUMRESLANGPROC lpEnumFunc, _In_ IParam)                                      |
| tssttuiuhi | BOOL | EnumResourceLanguagesEx(_In_ HMODULE _In_ LPCTSTR lpType, _In_ LPCTSTR lpName, ENUMRESLANGPROC lpEnumFunc, _In_ IParam, _In_ DWORD dwFlags, _In_ LANGID)  |
| taattuiuhi | BOOL | EnumResourceLanguagesExA(_In_ HMODULE _In_ LPCSTR lpType, _In_ LPCSTR lpName, ENUMRESLANGPROC lpEnumFunc, _In_ IParam, _In_ DWORD dwFlags, _In_ LANGID)   |
| twwttuiuhi | BOOL | EnumResourceLanguagesExW(_In_ HMODULE _In_ LPCWSTR lpType, _In_ LPCWSTR lpName, ENUMRESLANGPROC lpEnumFunc, _In_ IParam, _In_ DWORD dwFlags, _In_ LANGID) |
| twwtti     | BOOL | EnumResourceLanguagesW(_In_ HMODULE _In_ LPCWSTR lpType, _In_ LPCWSTR lpName, ENUMRESLANGPROC lpEnumFunc, _In_ IParam)                                    |
| tstti      | BOOL | EnumResourceName(_In_opt_ HMODULE LPCTSTR lpszType, _In_ ENUMRESNAMEPROC lpEnumFunc, _In_ LONG_PTR IParam)                                                |
|            |      |                                                                                                                                                           |

|           |      |                                                                                                                                                         |
|-----------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| tatti     | BOOL | EnumResourceNamesA(_In_opt_ HMODULE<br>_In_ LPCSTR lpszType, _In_ ENUMRESNA<br>lpEnumFunc, _In_ LONG_PTR lParam)                                        |
| tsttuiuhi | BOOL | EnumResourceNamesEx(_In_opt_ HMODULE<br>_In_ LPCTSTR lpszType, _In_ ENUMRESNA<br>lpEnumFunc, _In_ LONG_PTR lParam, _In_<br>dwFlags, _In_ LANGID LangId) |
| tattuiuhi | BOOL | EnumResourceNamesExA(_In_opt_ HMOD<br>_In_ LPCSTR lpszType, _In_ ENUMRESNA<br>lpEnumFunc, _In_ LONG_PTR lParam, _In_<br>dwFlags, _In_ LANGID LangId)    |
| twttuiuhi | BOOL | EnumResourceNamesExW(_In_opt_ HMOD<br>_In_ LPCWSTR lpszType, _In_ ENUMRESNA<br>lpEnumFunc, _In_ LONG_PTR lParam, _In_<br>dwFlags, _In_ LANGID LangId)   |
| twtti     | BOOL | EnumResourceNamesW(_In_opt_ HMODULE<br>_In_ LPCWSTR lpszType, _In_ ENUMRESNA<br>lpEnumFunc, _In_ LONG_PTR lParam)                                       |
| ttti      | BOOL | EnumResourceTypes(_In_opt_ HMODULE<br>_In_ ENUMRESTYPEPROC lpEnumFunc, _In_<br>lParam)                                                                  |
| ttti      | BOOL | EnumResourceTypesA(_In_opt_ HMODULE<br>_In_ ENUMRESTYPEPROC lpEnumFunc, _In_<br>lParam)                                                                 |
| tttuiuhi  | BOOL | EnumResourceTypesEx(_In_opt_ HMODULE<br>_In_ ENUMRESTYPEPROC lpEnumFunc, _In_<br>LONG_PTR lParam, _In_ DWORD dwFlags<br>LangId)                         |
| tttuiuhi  | BOOL | EnumResourceTypesExA(_In_opt_ HMODU<br>_In_ ENUMRESTYPEPROC lpEnumFunc, _In_<br>LONG_PTR lParam, _In_ DWORD dwFlags<br>LangId)                          |
| tttuiuhi  | BOOL | EnumResourceTypesExW(_In_opt_ HMODU<br>_In_ ENUMRESTYPEPROC lpEnumFunc, _In_<br>LONG_PTR lParam, _In_ DWORD dwFlags<br>LangId)                          |
| ttti      | BOOL | EnumResourceTypesW(_In_opt_ HMODULE<br>_In_ ENUMRESTYPEPROC lpEnumFunc, _In_<br>lParam)                                                                 |
| tuii      | BOOL | EnumSystemCodePages(_In_ CODEPAGE_<br>lpCodePageEnumProc, _In_ DWORD dwFla                                                                              |
| tuii      | BOOL | EnumSystemCodePagesA(_In_ CODEPAGE<br>lpCodePageEnumProc, _In_ DWORD dwFla                                                                              |
| tuii      | BOOL | EnumSystemCodePagesW(_In_ CODEPAGE<br>lpCodePageEnumProc, _In_ DWORD dwFla                                                                              |

|         |      |                                                                                                                                     |
|---------|------|-------------------------------------------------------------------------------------------------------------------------------------|
| uituiui | UINT | EnumSystemFirmwareTables(_In_ DWORD FirmwareTableProviderSignature, _Out_ PVOID FirmwareTableBuffer, _In_ DWORD BufferSize)         |
| uiuiti  | BOOL | EnumSystemGeoID(_In_ GEOCLASS GeoClass, _In_ GEOID ParentGeoId, _In_ GEO_ENUMPROC lpGeoEnumProc)                                    |
| tuiti   | BOOL | EnumSystemLanguageGroup(_In_ LANGUAGEGROUP_ENUMPROC lpLanguageGroupEnumProc, _In_ DWORD LONG_PTR lParam)                            |
| tuiti   | BOOL | EnumSystemLanguageGroupsA(_In_ LANGUAGEGROUP_ENUMPROC lpLanguageGroupEnumProc, _In_ DWORD LONG_PTR lParam)                          |
| tuiti   | BOOL | EnumSystemLanguageGroupsW(_In_ LANGUAGEGROUP_ENUMPROC lpLanguageGroupEnumProc, _In_ DWORD LONG_PTR lParam)                          |
| tuii    | BOOL | EnumSystemLocales(_In_ LOCALE_ENUMPROC lpLocaleEnumProc, _In_ DWORD dwFlags)                                                        |
| tuii    | BOOL | EnumSystemLocalesA(_In_ LOCALE_ENUMPROC lpLocaleEnumProc, _In_ DWORD dwFlags)                                                       |
| tuitti  | BOOL | EnumSystemLocalesEx(_In_ LOCALE_ENUMPROC lpLocaleEnumProcEx, _In_ DWORD dwFlags, LPARAM lParam, _In_opt_ LPVOID lpReserved)         |
| tuii    | BOOL | EnumSystemLocalesW(_In_ LOCALE_ENUMPROC lpLocaleEnumProc, _In_ DWORD dwFlags)                                                       |
| tuiiii  | BOOL | EnumTimeFormat(_In_ TIMEFMT_ENUMPROC lpTimeFmtEnumProc, _In_ LCID Locale, _In_ DWORD dwFlags)                                       |
| tuiiii  | BOOL | EnumTimeFormatsA(_In_ TIMEFMT_ENUMPROC lpTimeFmtEnumProc, _In_ LCID Locale, _In_ DWORD dwFlags)                                     |
| twuiti  | BOOL | EnumTimeFormatsEx(_In_ TIMEFMT_ENUMPROC lpTimeFmtEnumProcEx, _In_opt_ LPCWSTR lpLocaleName, _In_ DWORD dwFlags, _In_ LPARAM lParam) |
| tuiiii  | BOOL | EnumTimeFormatsW(_In_ TIMEFMT_ENUMPROC lpTimeFmtEnumProc, _In_ LCID Locale, _In_ DWORD dwFlags)                                     |
| tuiti   | BOOL | EnumUILanguage(_In_ UILANGUAGE_ENUMPROC lpUILanguageEnumProc, _In_ DWORD dwFlags, _In_ LPARAM lParam)                               |
| tuiti   | BOOL | EnumUILanguagesA(_In_ UILANGUAGE_ENUMPROC lpUILanguageEnumProc, _In_ DWORD dwFlags, _In_ LPARAM lParam)                             |

|           |       |                                                                                                                                                                   |
|-----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |       | LONG_PTR lParam)                                                                                                                                                  |
| tuiti     | BOOL  | EnumUILanguagesW(_In_ UI_LANGUAGE_ENUM dwLanguageId, _In_ lpUILanguageEnumProc, _In_ DWORD dwFlags, _In_ LONG_PTR lParam)                                         |
| tuiiui    | DWORD | EraseTape(_In_ HANDLE hDevice, _In_ DWORD dwEraseType, _In_ BOOL bImmediate)                                                                                      |
| tuii      | BOOL  | EscapeCommFunction(_In_ HANDLE hFile, _In_ DWORD dwFunc)                                                                                                          |
| uii       | VOID  | ExitProcess(_In_ UINT uExitCode)                                                                                                                                  |
| uii       | VOID  | ExitThread(_In_ DWORD dwExitCode)                                                                                                                                 |
| ssuiui    | DWORD | ExpandEnvironmentStrings(_In_ LPCTSTR lpSrc, _Out_ LPTSTR lpDst, _In_ DWORD nSize)                                                                                |
| aauiui    | DWORD | ExpandEnvironmentStringsA(_In_ LPCSTR lpSrc, _Out_opt_ LPSTR lpDst, _In_ DWORD nSize)                                                                             |
| wwuiui    | DWORD | ExpandEnvironmentStringsW(_In_ LPCWSTR lpSrc, _Out_opt_ LPWSTR lpDst, _In_ DWORD nSize)                                                                           |
| uisi      | VOID  | FatalAppExit(_In_ UINT uAction, _In_ LPCWSTR lpMessageText)                                                                                                       |
| uiai      | VOID  | FatalAppExitA(_In_ UINT uAction, _In_ LPCSTR lpMessageText)                                                                                                       |
| uiwi      | VOID  | FatalAppExitW(_In_ UINT uAction, _In_ LPCWSTR lpMessageText)                                                                                                      |
| ii        | VOID  | FatalExit(_In_ int ExitCode)                                                                                                                                      |
| ttti      | BOOL  | FileTimeToDosDateTime(_In_ const FILETIME *lpFileTime, _Out_ LPWORD lpFatDate, _Out_ LPWORD lpFatTime)                                                            |
| tti       | BOOL  | FileTimeToLocalFileTime(_In_ const FILETIME *lpFileTime, _Out_ LPFILETIME lpLocalFileTime)                                                                        |
| tti       | BOOL  | FileTimeToSystemTime(_In_ const FILETIME *lpFileTime, _Out_ LPSYSTEMTIME lpSystemTime)                                                                            |
| tuhuiuiti | BOOL  | FillConsoleOutputAttribute(_In_ HANDLE hConsoleOutput, _In_ WORD wAttribute, _In_ DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfAttrsWritten)   |
| tyuiuiti  | BOOL  | FillConsoleOutputCharacter(_In_ HANDLE hConsoleOutput, _In_ TCHAR cCharacter, _In_ DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfCharsWritten)  |
| tyuiuiti  | BOOL  | FillConsoleOutputCharacterA(_In_ HANDLE hConsoleOutput, _In_ TCHAR cCharacter, _In_ DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfCharsWritten) |
|           |       |                                                                                                                                                                   |

|            |        |                                                                                                                                                                                                                                    |
|------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tyuiuiti   | BOOL   | FindConsoleOutputCharacterW(_In_ HANDLE hConsoleOutput, _In_ TCHAR cCharacter, _In_ ULONG nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfCharsWritten)                                                                  |
| uituitti   | BOOL   | FindActCtxSectionGuid(_In_ DWORD dwFlags, _In_ const GUID *lpExtensionGuid, _In_ ULONG ulSectionIndex, _Out_ const GUID *lpGuidToFind, _Out_ PACTCTX_SECTION_KEYED_DATA ReturnData)                                                |
| uituisti   | BOOL   | FindActCtxSectionString(_In_ DWORD dwFlags, _In_ const GUID *lpExtensionGuid, _In_ ULONG ulSectionIndex, _In_ LPCTSTR lpStringToFind, _Out_ LPCTSTR lpString, _Out_ PACTCTX_SECTION_KEYED_DATA ReturnData)                         |
| uituiati   | BOOL   | FindActCtxSectionStringA(_In_ DWORD dwFlags, _In_ const GUID *lpExtensionGuid, _In_ ULONG ulSectionIndex, _In_ LPCSTR lpStringToFind, _Out_ LPCSTR lpString, _Out_ PACTCTX_SECTION_KEYED_DATA ReturnData)                          |
| uituiwti   | BOOL   | FindActCtxSectionStringW(_In_ DWORD dwFlags, _In_ const GUID *lpExtensionGuid, _In_ ULONG ulSectionIndex, _In_ LPCWSTR lpStringToFind, _Out_ LPCWSTR lpString, _Out_ PACTCTX_SECTION_KEYED_DATA ReturnData)                        |
| suh        | ATOM   | FindAtom(_In_ LPCTSTR lpString)                                                                                                                                                                                                    |
| auh        | ATOM   | FindAtomA(_In_ LPCSTR lpString)                                                                                                                                                                                                    |
| wuh        | ATOM   | FindAtomW(_In_ LPCWSTR lpString)                                                                                                                                                                                                   |
| ti         | BOOL   | FindClose(_Inout_ HANDLE hFindFile)                                                                                                                                                                                                |
| ti         | BOOL   | FindCloseChangeNotification(_In_ HANDLE hChangeHandle)                                                                                                                                                                             |
| siuit      | HANDLE | FindFirstChangeNotification(_In_ LPCTSTR lpPathName, _In_ BOOL bWatchSubtree, _In_ DWORD dwNotifyFilter)                                                                                                                           |
| aiuit      | HANDLE | FindFirstChangeNotificationA(_In_ LPCSTR lpPathName, _In_ BOOL bWatchSubtree, _In_ DWORD dwNotifyFilter)                                                                                                                           |
| wiuit      | HANDLE | FindFirstChangeNotificationW(_In_ LPCWSTR lpPathName, _In_ BOOL bWatchSubtree, _In_ DWORD dwNotifyFilter)                                                                                                                          |
| stt        | HANDLE | FindFirstFile(_In_ LPCTSTR lpFileName, _Out_ LPWIN32_FIND_DATA lpFindFileData)                                                                                                                                                     |
| att        | HANDLE | FindFirstFileA(_In_ LPCSTR lpFileName, _Out_ LPWIN32_FIND_DATA lpFindFileData)                                                                                                                                                     |
| suituituit | HANDLE | FindFirstFileEx(_In_ LPCTSTR lpFileName, _In_ FINDEX_INFO_LEVELS fInfoLevelId, _Out_ LPWIN32_FIND_DATA lpFindFileData, _In_ FINDEX_SEARCH_OPTIONS dwSearchOptions, _Reserved_ LPVOID lpSearchFilter, _In_ DWORD dwAdditionalFlags) |
|            |        | FindFirstFileExA(_In_ LPCSTR lpFileName, _In_ FINDEX_INFO_LEVELS fInfoLevelId, _Out_ LPWIN32_FIND_DATA lpFindFileData, _In_ FINDEX_SEARCH_OPTIONS dwSearchOptions, _Reserved_ LPVOID lpSearchFilter, _In_ DWORD dwAdditionalFlags) |

|            |        |                                                                                                                                                                                                                  |
|------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aituituit  | HANDLE | lpFindFileData, _In_ FINDEX_SEARCH_O<br>_Reserved_ LPVOID lpSearchFilter, _In_ D<br>dwAdditionalFlags)                                                                                                           |
| wuituituit | HANDLE | FindFirstFileExW(_In_ LPCWSTR lpFileNa<br>FINDEX_INFO_LEVELS fInfoLevelId, _Ou<br>lpFindFileData, _In_ FINDEX_SEARCH_O<br>_Reserved_ LPVOID lpSearchFilter, _In_ D<br>dwAdditionalFlags)                         |
| wuitwtt    | HANDLE | FindFirstFileNameTransactedW(_In_ LPCW<br>lpFileName, _In_ DWORD dwFlags, _Inout_<br>StringLength, _Inout_ PWCHAR LinkName<br>HANDLE hTransaction)                                                               |
| wuitwt     | HANDLE | FindFirstFileNameW(_In_ LPCWSTR lpFile<br>DWORD dwFlags, _Inout_ LPDWORD Stri<br>_Inout_ PWCHAR LinkName)                                                                                                        |
| suituituit | HANDLE | FindFirstFileTransacted(_In_ LPCTSTR lpFi<br>FINDEX_INFO_LEVELS fInfoLevelId, _Ou<br>lpFindFileData, _In_ FINDEX_SEARCH_O<br>_Reserved_ LPVOID lpSearchFilter, _In_ D<br>dwAdditionalFlags, _In_ HANDLE hTransac |
| aituituit  | HANDLE | FindFirstFileTransactedA(_In_ LPCSTR lpF<br>FINDEX_INFO_LEVELS fInfoLevelId, _Ou<br>lpFindFileData, _In_ FINDEX_SEARCH_O<br>_Reserved_ LPVOID lpSearchFilter, _In_ D<br>dwAdditionalFlags, _In_ HANDLE hTransac  |
| wuituituit | HANDLE | FindFirstFileTransactedW(_In_ LPCWSTR l<br>FINDEX_INFO_LEVELS fInfoLevelId, _Ou<br>lpFindFileData, _In_ FINDEX_SEARCH_O<br>_Reserved_ LPVOID lpSearchFilter, _In_ D<br>dwAdditionalFlags, _In_ HANDLE hTransac   |
| wtt        | HANDLE | FindFirstFileW(_In_ LPCWSTR lpFileName<br>LPWIN32_FIND_DATA lpFindFileData)                                                                                                                                      |
| wuituit    | HANDLE | FindFirstStreamTransactedW(_In_ LPCWST<br>_In_ STREAM_INFO_LEVELS InfoLevel, _<br>lpFindStreamData, _Reserved_ DWORD dw<br>HANDLE hTransaction)                                                                  |
| wuituit    | HANDLE | FindFirstStreamW(_In_ LPCWSTR lpFileNa<br>STREAM_INFO_LEVELS InfoLevel, _Out_<br>lpFindStreamData, _Reserved_ DWORD dw                                                                                           |
| suit       | HANDLE | FindFirstVolume(_Out_ LPTSTR lpszVolum<br>DWORD cchBufferLength)                                                                                                                                                 |
| ait        | HANDLE | FindFirstVolumeA(_Out_ LPSTR lpszVolum<br>DWORD cchBufferLength)                                                                                                                                                 |
| ssuit      | HANDLE | FindFirstVolumeMountPoint(_In_ LPTSTR<br>lpszRootPathName, _Out_ LPTSTR lpszVolu<br>_In_ DWORD cchBufferLength)                                                                                                  |

|             |        |                                                                                                                                                                                             |
|-------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aauiit      | HANDLE | FindFirstVolumeMountPointA(_In_ LPSTR lpszRootPathName, _Out_ LPSTR lpszVolumeMountPoint, _In_ DWORD cchBufferLength)                                                                       |
| wwuiit      | HANDLE | FindFirstVolumeMountPointW(_In_ LPWSTR lpszRootPathName, _Out_ LPWSTR lpszVolumeMountPoint, _In_ DWORD cchBufferLength)                                                                     |
| wuiit       | HANDLE | FindFirstVolumeW(_Out_ LPWSTR lpszVolumeMountPoint, _In_ DWORD cchBufferLength)                                                                                                             |
| ti          | BOOL   | FindNextChangeNotification(_In_ HANDLE hChangeHandle)                                                                                                                                       |
| tii         | BOOL   | FindNextFile(_In_ HANDLE hFindFile, _Out_ LPWIN32_FIND_DATA lpFindFileData)                                                                                                                 |
| tii         | BOOL   | FindNextFileA(_In_ HANDLE hFindFile, _Out_ LPWIN32_FIND_DATA lpFindFileData)                                                                                                                |
| ttwi        | BOOL   | FindNextFileNameW(_In_ HANDLE hFindFile, _Out_ LPWSTR lpszFileName, _In_ LPDWORD StringLength, _Inout_ PWCHAR lpFileName)                                                                   |
| tii         | BOOL   | FindNextFileW(_In_ HANDLE hFindFile, _Out_ LPWIN32_FIND_DATA lpFindFileData)                                                                                                                |
| tii         | BOOL   | FindNextStreamW(_In_ HANDLE hFindStream, _Out_ LPVOID lpFindStreamData)                                                                                                                     |
| tsuii       | BOOL   | FindNextVolume(_In_ HANDLE hFindVolume, _Out_ LPTSTR lpszVolumeName, _In_ DWORD cchBufferLength)                                                                                            |
| tauii       | BOOL   | FindNextVolumeA(_In_ HANDLE hFindVolume, _Out_ LPSTR lpszVolumeName, _In_ DWORD cchBufferLength)                                                                                            |
| tsuii       | BOOL   | FindNextVolumeMountPoint(_In_ HANDLE hFindVolumeMountPoint, _Out_ LPTSTR lpszVolumeMountPoint, _In_ DWORD cchBufferLength)                                                                  |
| tauii       | BOOL   | FindNextVolumeMountPointA(_In_ HANDLE hFindVolumeMountPoint, _Out_ LPSTR lpszVolumeMountPoint, _In_ DWORD cchBufferLength)                                                                  |
| twuii       | BOOL   | FindNextVolumeMountPointW(_In_ HANDLE hFindVolumeMountPoint, _Out_ LPWSTR lpszVolumeMountPoint, _In_ DWORD cchBufferLength)                                                                 |
| twuii       | BOOL   | FindNextVolumeW(_In_ HANDLE hFindVolume, _Out_ LPWSTR lpszVolumeName, _In_ DWORD cchBufferLength)                                                                                           |
| uiuiwiwiti  | int    | FindNLSString(_In_ LCID Locale, _In_ DWORD dwFindNLSStringFlags, _In_ LPCWSTR lpStringSource, _In_ int cchSource, _In_ LPCWSTR lpStringValue, _In_ int cchValue, _Out_opt_ LPINT pcchFound) |
| wuiwiwittti | int    | FindNLSStringEx(_In_opt_ LPCWSTR lpStringSource, _In_ int cchSource, _In_ LPCWSTR lpStringValue, _In_ int cchValue, _Out_opt_ LPINT pcchFound)                                              |

|          |       |                                                                                                                                                             |
|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |       | pcchFound, _In_opt_ LPNLSVERSIONINFO lpVersionInformation, _In_opt_ LPVOID lpFindResource, _In_opt_ LPARAM sortHandle)                                      |
| tsst     | HRSRC | FindResource(_In_opt_ HMODULE hModule, LPCWSTR lpName, _In_ LPCWSTR lpType)                                                                                 |
| taat     | HRSRC | FindResourceA(_In_opt_ HMODULE hModule, LPCSTR lpName, _In_ LPCSTR lpType)                                                                                  |
| tssuht   | HRSRC | FindResourceEx(_In_opt_ HMODULE hModule, LPCWSTR lpType, _In_ LPCWSTR lpName, wLanguage)                                                                    |
| taauht   | HRSRC | FindResourceExA(_In_opt_ HMODULE hModule, LPCSTR lpType, _In_ LPCSTR lpName, wLanguage)                                                                     |
| twwuht   | HRSRC | FindResourceExW(_In_opt_ HMODULE hModule, LPCWSTR lpType, _In_ LPCWSTR lpName, wLanguage)                                                                   |
| twwt     | HRSRC | FindResourceW(_In_opt_ HMODULE hModule, LPCWSTR lpName, _In_ LPCWSTR lpType)                                                                                |
| uiwiwiii | int   | FindStringOrdinal(_In_ DWORD dwFindString, _In_ LPCWSTR lpStringSource, _In_ int cchFindString, _In_ LPCWSTR lpStringValue, _In_ int cchValue, bIgnoreCase) |
| ti       | BOOL  | FindVolumeClose(_In_ HANDLE hFindVolume)                                                                                                                    |
| ti       | BOOL  | FindVolumeMountPointClose(_In_ HANDLE hFindVolumeMountPoint)                                                                                                |
| tui      | DWORD | FlsAlloc(_In_ PFLS_CALLBACK_FUNCTION)                                                                                                                       |
| uii      | BOOL  | FlsFree(_In_ DWORD dwFlsIndex)                                                                                                                              |
| uit      | PVOID | FlsGetValue(_In_ DWORD dwFlsIndex)                                                                                                                          |
| uiti     | BOOL  | FlsSetValue(_In_ DWORD dwFlsIndex, _In_ lpFlsData)                                                                                                          |
| ti       | BOOL  | FlushConsoleInputBuffer(_In_ HANDLE hConsole)                                                                                                               |
| ti       | BOOL  | FlushFileBuffers(_In_ HANDLE hFile)                                                                                                                         |
| ttti     | BOOL  | FlushInstructionCache(_In_ HANDLE hProcess, LPCVOID lpBaseAddress, _In_ SIZE_T dwSize)                                                                      |
| i        | VOID  | FlushProcessWriteBuffer(void)                                                                                                                               |
| titi     | BOOL  | FlushViewOfFile(_In_ LPCVOID lpBaseAddress, SIZE_T dwNumberOfBytesToFlush)                                                                                  |
| uisisii  | int   | FoldString(_In_ DWORD dwMapFlags, _In_ lpSrcStr, _In_ int cchSrc, _Out_opt_ LPTSTR lpDest, int cchDest)                                                     |
| uiaiaii  | int   | FoldStringA(_In_ DWORD dwMapFlags, _In_ lpSrcStr, _In_ int cchSrc, _Out_opt_ LPSTR lpDest, int cchDest)                                                     |

|               |         |                                                                                                                                                                                                      |
|---------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |         | int cchDest)                                                                                                                                                                                         |
| uiwiwii       | int     | FoldStringW(_In_ DWORD dwMapFlags, _In_ LPCTSTR lpSrcStr, _In_ int cchSrc, _Out_opt_ LPWSTR lpDestStr, _In_ int cchDest)                                                                             |
| uituiuisuitui | DWORD   | FormatMessage(_In_ DWORD dwFlags, _In_ LPCVOID lpSource, _In_ DWORD dwMessageId, _In_ DWORD dwLanguageId, _Out_ LPTSTR lpBuffer, _In_ int nSize, _In_opt_ va_list *Arguments)                        |
| uituiuiauitui | DWORD   | FormatMessageA(_In_ DWORD dwFlags, _In_ LPCVOID lpSource, _In_ DWORD dwMessageId, _In_ DWORD dwLanguageId, _Out_ LPSTR lpBuffer, _In_ int nSize, _In_opt_ va_list *Arguments)                        |
| uituiuiwuitui | DWORD   | FormatMessageW(_In_ DWORD dwFlags, _In_ LPCVOID lpSource, _In_ DWORD dwMessageId, _In_ DWORD dwLanguageId, _Out_ LPWSTR lpBuffer, _In_ int nSize, _In_opt_ va_list *Arguments)                       |
| i             | BOOL    | FreeConsole(void)                                                                                                                                                                                    |
| si            | BOOL    | FreeEnvironmentStrings(_In_ LPTCH lpzEnvironmentBlock)                                                                                                                                               |
| ai            | BOOL    | FreeEnvironmentStringsA(_In_ LPCH lpzEnvironmentBlock)                                                                                                                                               |
| wi            | BOOL    | FreeEnvironmentStringsW(_In_ LPWCH lpzEnvironmentBlock)                                                                                                                                              |
| ti            | BOOL    | FreeLibrary(_In_ HMODULE hModule)                                                                                                                                                                    |
| tuii          | VOID    | FreeLibraryAndExitThread(_In_ HMODULE hModule, DWORD dwExitCode)                                                                                                                                     |
| tui           | VOID    | FreeLibraryWhenCallbackReturns(_Inout_ PTP_CALLBACK_INSTANCE pci, _In_ HMODULE hModule)                                                                                                              |
| ti            | BOOL    | FreeResource(_In_ HGLOBAL hglbResource)                                                                                                                                                              |
| ttti          | BOOL    | FreeUsesPhysicalPages(_In_ HANDLE hProcess, _In_ PULONG_PTR NumberOfPages, _In_ PULONG_PTR UserPfnArray)                                                                                             |
| uiiii         | BOOL    | GenerateConsoleCtrlEvent(_In_ DWORD dwCtrlEvent, _In_ DWORD dwProcessGroupId)                                                                                                                        |
| ui            | UINT    | GetACP(void)                                                                                                                                                                                         |
| ui            | UINT    | GetACP(void)                                                                                                                                                                                         |
| uhui          | DWORD   | GetActiveProcessorCount(_In_ WORD GroupNumber)                                                                                                                                                       |
| uh            | WORD    | GetActiveProcessorGroupCount(void)                                                                                                                                                                   |
| tttiti        | HRESULT | GetApplicationRecoveryCallback(_In_ HANDLE hProcess, _Out_ APPLICATION_RECOVERY_CALLBACK *pRecoveryCallback, _Out_ PVOID *ppvParameters, _Out_ PDWORD pdwPingInterval, _Out_ PDWORD pdwPingInterval) |

|            |         |                                                                                                                                                                                               |
|------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttti       | HRESULT | GetApplicationRestartSettings(_In_ HANDLE hProcess, _Out_opt_ PWSTR pwzCommandline, _Inout_ DWORD dwFlags, _Out_opt_ PDWORD pdwFlags)                                                         |
| uhsiui     | UINT    | GetAtomName(_In_ ATOM nAtom, _Out_ LPWSTR lpBuffer, _In_ int nSize)                                                                                                                           |
| uhaiui     | UINT    | GetAtomNameA(_In_ ATOM nAtom, _Out_ LPSTR lpBuffer, _In_ int nSize)                                                                                                                           |
| uhwiui     | UINT    | GetAtomNameW(_In_ ATOM nAtom, _Out_ LPWSTR lpBuffer, _In_ int nSize)                                                                                                                          |
| sti        | BOOL    | GetBinaryType(_In_ LPCTSTR lpApplicationName, _Out_ LPDWORD lpBinaryType)                                                                                                                     |
| ati        | BOOL    | GetBinaryTypeA(_In_ LPCSTR lpApplicationName, _Out_ LPDWORD lpBinaryType)                                                                                                                     |
| wti        | BOOL    | GetBinaryTypeW(_In_ LPCWSTR lpApplicationName, _Out_ LPDWORD lpBinaryType)                                                                                                                    |
| wuitwii    | BOOL    | GetCalendarDateFormatEx(_In_ LPCWSTR lpCalendar, _In_ DWORD dwFlags, _In_ const LPCALDATE lpCalDateTime, _In_ LPCWSTR lpFormat, _Out_ LPWSTR lpDateStr, _In_ int cchDate)                     |
| uiuiuisiti | int     | GetCalendarInfo(_In_ LCID Locale, _In_ CALTYPE CalType, _Out_opt_ LPTSTR lpDateStr, _In_ int cchData, _Out_opt_ LPDWORD lpValue)                                                              |
| uiuiuiaiti | int     | GetCalendarInfoA(_In_ LCID Locale, _In_ CALTYPE CalType, _Out_opt_ LPSTR lpDateStr, _In_ int cchData, _Out_opt_ LPDWORD lpValue)                                                              |
| wuiwuiwiti | int     | GetCalendarInfoEx(_In_opt_ LPCWSTR lpCalendar, _In_ CALID Calendar, _In_opt_ LPCWSTR lpFormat, _In_ CALTYPE CalType, _Out_opt_ LPWSTR lpDateStr, _In_ int cchData, _Out_opt_ LPDWORD lpValue) |
| uiuiuiwiti | int     | GetCalendarInfoW(_In_ LCID Locale, _In_ CALTYPE CalType, _Out_opt_ LPWSTR lpDateStr, _In_ int cchData, _Out_opt_ LPDWORD lpValue)                                                             |
| uitti      | BOOL    | GetCalendarSupportedDateRange(_In_ CALTYPE CalType, _Out_ LPCALDATETIME lpCalMinDateTime, _Out_ LPCALDATETIME lpCalMaxDateTime)                                                               |
| s          | LPTSTR  | GetCommandLine(void)                                                                                                                                                                          |
| a          | LPSTR   | GetCommandLineA(void)                                                                                                                                                                         |
| w          | LPWSTR  | GetCommandLineW(void)                                                                                                                                                                         |
| ttti       | BOOL    | GetCommConfig(_In_ HANDLE hCommDev, _Out_ LPCOMMCONFIG lpCC, _Inout_ LPDWORD pdwFlags)                                                                                                        |
| tii        | BOOL    | GetCommMask(_In_ HANDLE hFile, _Out_ LPDWORD lpEvtMask)                                                                                                                                       |
|            |         | GetCommModemStatus(_In_ HANDLE hFile, _Out_ LPDWORD lpModemStatus)                                                                                                                            |

|         |       |                                                                                                            |
|---------|-------|------------------------------------------------------------------------------------------------------------|
| ttd     | BOOL  | LPDWORD lpModemStat)                                                                                       |
| ttd     | BOOL  | GetCommProperties(_In_ HANDLE hFile, _<br>LPCOMMPROP lpCommProp)                                           |
| ttd     | BOOL  | GetCommState(_In_ HANDLE hFile, _Inout<br>lpDCB)                                                           |
| ttd     | BOOL  | GetCommTimeouts(_In_ HANDLE hFile, _C<br>LPCOMMTIMEOUTS lpCommTimeouts)                                    |
| stui    | DWORD | GetCompressedFileSize(_In_ LPCTSTR lpF<br>_Out_opt_ LPDWORD lpFileSizeHigh)                                |
| atui    | DWORD | GetCompressedFileSizeA(_In_ LPCSTR lpF<br>_Out_opt_ LPDWORD lpFileSizeHigh)                                |
| sttui   | DWORD | GetCompressedFileSizeTransacted(_In_ LPC<br>lpFileName, _Out_opt_ LPDWORD lpFileSi<br>HANDLE hTransaction) |
| attui   | DWORD | GetCompressedFileSizeTransactedA(_In_ LP<br>lpFileName, _Out_opt_ LPDWORD lpFileSi<br>HANDLE hTransaction) |
| wttui   | DWORD | GetCompressedFileSizeTransactedW(_In_ LP<br>lpFileName, _Out_opt_ LPDWORD lpFileSi<br>HANDLE hTransaction) |
| wtui    | DWORD | GetCompressedFileSizeW(_In_ LPCWSTR l<br>_Out_opt_ LPDWORD lpFileSizeHigh)                                 |
| sti     | BOOL  | GetComputerName(_Out_ LPTSTR lpBuffer<br>LPDWORD lpnSize)                                                  |
| ati     | BOOL  | GetComputerNameA(_Out_ LPSTR lpBuffer<br>LPDWORD lpnSize)                                                  |
| uisti   | BOOL  | GetComputerNameEx(_In_ COMPUTER_N<br>NameType, _Out_ LPTSTR lpBuffer, _Inout<br>lpnSize)                   |
| uiati   | BOOL  | GetComputerNameExA(_In_<br>COMPUTER_NAME_FORMAT NameType<br>lpBuffer, _Inout_ LPDWORD lpnSize)             |
| uiwti   | BOOL  | GetComputerNameExW(_In_<br>COMPUTER_NAME_FORMAT NameType<br>LPWSTR lpBuffer, _Inout_ LPDWORD lpnS          |
| wti     | BOOL  | GetComputerNameW(_Out_ LPWSTR lpBu<br>LPDWORD lpnSize)                                                     |
| ssuisui | DWORD | GetConsoleAlias(_In_ LPTSTR lpSource, _C<br>lpTargetBuffer, _In_ DWORD TargetBufferL<br>LPTSTR lpExeName)  |
| aauiui  | DWORD | GetConsoleAliasA(_In_ LPSTR lpSource, _C<br>lpTargetBuffer, _In_ DWORD TargetBufferL<br>LPSTR lpExeName)   |

|         |       |                                                                                                          |
|---------|-------|----------------------------------------------------------------------------------------------------------|
| suisui  | DWORD | GetConsoleAliases(_Out_ LPTSTR lpAliasB<br>DWORD AliasBufferLength, _In_ LPTSTR                          |
| auiaui  | DWORD | GetConsoleAliasesA(_Out_ LPSTR lpAliasE<br>DWORD AliasBufferLength, _In_ LPSTR lp                        |
| sui     | DWORD | GetConsoleAliasesLength(_In_ LPTSTR lpE                                                                  |
| aui     | DWORD | GetConsoleAliasesLengthA(_In_ LPSTR lpE                                                                  |
| wui     | DWORD | GetConsoleAliasesLengthW(_In_ LPWSTR                                                                     |
| wuiwui  | DWORD | GetConsoleAliasesW(_Out_ LPWSTR lpAli<br>DWORD AliasBufferLength, _In_ LPWSTR                            |
| suiui   | DWORD | GetConsoleAliasExes(_Out_ LPTSTR lpExe<br>_In_ DWORD ExeNameBufferLength)                                |
| auiui   | DWORD | GetConsoleAliasExesA(_Out_ LPSTR lpExe<br>_In_ DWORD ExeNameBufferLength)                                |
| ui      | DWORD | GetConsoleAliasExesLength(void)                                                                          |
| ui      | DWORD | GetConsoleAliasExesLengthA(void)                                                                         |
| ui      | DWORD | GetConsoleAliasExesLengthW(void)                                                                         |
| wuiui   | DWORD | GetConsoleAliasExesW(_Out_ LPWSTR lpE<br>_In_ DWORD ExeNameBufferLength)                                 |
| wwuiwui | DWORD | GetConsoleAliasW(_In_ LPWSTR lpSource<br>LPWSTR lpTargetBuffer, _In_ DWORD Tar<br>_In_ LPWSTR lpExeName) |
| ui      | UINT  | GetConsoleCP(void)                                                                                       |
| ui      | UINT  | GetConsoleCP(void)                                                                                       |
| tti     | BOOL  | GetConsoleCursorInfo(_In_ HANDLE hCon<br>_Out_ PCONSOLE_CURSOR_INFO lpCon                                |
| ti      | BOOL  | GetConsoleDisplayMode(_Out_ LPDWORD                                                                      |
| tuiui   | COORD | GetConsoleFontSize(_In_ HANDLE hConso<br>DWORD nFont)                                                    |
| ti      | BOOL  | GetConsoleHistoryInfo(_Out_<br>PCONSOLE_HISTORY_INFO lpConsoleHi                                         |
| tti     | BOOL  | GetConsoleMode(_In_ HANDLE hConsoleF<br>LPDWORD lpMode)                                                  |
| suiui   | DWORD | GetConsoleOriginalTitle(_Out_ LPTSTR lpC<br>_In_ DWORD nSize)                                            |
| auiui   | DWORD | GetConsoleOriginalTitleA(_Out_ LPSTR lpC<br>_In_ DWORD nSize)                                            |
| wuiui   | DWORD | GetConsoleOriginalTitleW(_Out_ LPWSTR<br>_In_ DWORD nSize)                                               |
| ui      | UINT  | GetConsoleOutputCP(void)                                                                                 |
| ui      | UINT  | GetConsoleOutputCP(void)                                                                                 |

|            |       |                                                                                                                                                                       |
|------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiui      | DWORD | GetConsoleProcessList(_Out_ LPDWORD lpProcessList, _In_ DWORD dwProcessCount)                                                                                         |
| tii        | BOOL  | GetConsoleScreenBufferInfo(_In_ HANDLE hConsoleOutput, _Out_ PCONSOLE_SCREEN_BUFFER_INFO lpConsoleScreenBufferInfo)                                                   |
| tii        | BOOL  | GetConsoleScreenBufferInfoEx(_In_ HANDLE hConsoleOutput, _Out_ PCONSOLE_SCREEN_BUFFER_INFOEX lpConsoleScreenBufferInfoEx)                                             |
| ti         | BOOL  | GetConsoleSelectionInfo(_Out_ PCONSOLE_SELECTION_INFO lpConsoleSelectionInfo)                                                                                         |
| suiui      | DWORD | GetConsoleTitle(_Out_ LPTSTR lpConsoleTitle, DWORD nSize)                                                                                                             |
| auuii      | DWORD | GetConsoleTitleA(_Out_ LPSTR lpConsoleTitle, DWORD nSize)                                                                                                             |
| wuiui      | DWORD | GetConsoleTitleW(_Out_ LPWSTR lpConsoleTitle, DWORD nSize)                                                                                                            |
| t          | HWND  | GetConsoleWindow(void)                                                                                                                                                |
| uiti       | BOOL  | GetCPInfo(_In_ UINT CodePage, _Out_ LPCPI lpCPInfo)                                                                                                                   |
| uiuiti     | BOOL  | GetCPInfoEx(_In_ UINT CodePage, _In_ DWORD dwFlags, _Out_ LPCPINFEX lpCPInfoEx)                                                                                       |
| uiuiti     | BOOL  | GetCPInfoExA(_In_ UINT CodePage, _In_ DWORD dwFlags, _Out_ LPCPINFEX lpCPInfoEx)                                                                                      |
| uiuiti     | BOOL  | GetCPInfoExW(_In_ UINT CodePage, _In_ DWORD dwFlags, _Out_ LPCPINFEX lpCPInfoEx)                                                                                      |
| uiuiistsii | int   | GetCurrencyFormat(_In_ LCID Locale, _In_ DWORD dwFlags, _In_ LPCTSTR lpValue, _In_opt_ CURRENCYFMT *lpFormat, _Out_opt_ LPCTSTR lpCurrencyStr, _In_ int cchCurrency)  |
| uiuiataii  | int   | GetCurrencyFormatA(_In_ LCID Locale, _In_ DWORD dwFlags, _In_ LPCSTR lpValue, _In_opt_ CURRENCYFMT *lpFormat, _Out_opt_ LPCTSTR lpCurrencyStr, _In_ int cchCurrency)  |
| wuiwtwii   | int   | GetCurrencyFormatEx(_In_opt_ LPCWSTR lpValue, _In_ DWORD dwFlags, _In_ LPCWSTR lpFormat, _Out_opt_ LPCTSTR lpCurrencyStr, _In_ int cchCurrency)                       |
| uiuiwtwii  | int   | GetCurrencyFormatW(_In_ LCID Locale, _In_ DWORD dwFlags, _In_ LPCWSTR lpValue, _In_opt_ CURRENCYFMT *lpFormat, _Out_opt_ LPCTSTR lpCurrencyStr, _In_ int cchCurrency) |
| ti         | BOOL  | GetCurrentActCtx(_Out_ HANDLE *lpActCtx)                                                                                                                              |

|           |        |                                                                                                                                                                       |
|-----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| titi      | BOOL   | GetCurrentConsoleFont(_In_ HANDLE hCo<br>_In_ BOOL bMaximumWindow, _Out_<br>PCONSOLE_FONT_INFO lpConsoleCurren                                                        |
| titi      | BOOL   | GetCurrentConsoleFontEx(_In_ HANDLE h<br>_In_ BOOL bMaximumWindow, _Out_<br>PCONSOLE_FONT_INFOEX lpConsoleCu                                                          |
| uisui     | DWORD  | GetCurrentDirectory(_In_ DWORD nBuffer<br>LPTSTR lpBuffer)                                                                                                            |
| uiaui     | DWORD  | GetCurrentDirectoryA(_In_ DWORD nBuffe<br>LPSTR lpBuffer)                                                                                                             |
| uiwui     | DWORD  | GetCurrentDirectoryW(_In_ DWORD nBuff<br>LPWSTR lpBuffer)                                                                                                             |
| t         | HANDLE | GetCurrentProcess(void)                                                                                                                                               |
| ui        | DWORD  | GetCurrentProcessId(void)                                                                                                                                             |
| ui        | DWORD  | GetCurrentProcessorNumber(void)                                                                                                                                       |
| ti        | VOID   | GetCurrentProcessorNumberEx(_Out_<br>PPROCESSOR_NUMBER ProcNumber)                                                                                                    |
| t         | HANDLE | GetCurrentThread(void)                                                                                                                                                |
| ui        | DWORD  | GetCurrentThreadId(void)                                                                                                                                              |
| uiuitssii | int    | GetDateFormat(_In_ LCID Locale, _In_ DW<br>_In_opt_ const SYSTEMTIME *lpDate, _In_<br>lpFormat, _Out_opt_ LPTSTR lpDateStr, _In                                       |
| uiuitaaii | int    | GetDateFormatA(_In_ LCID Locale, _In_ D<br>_In_opt_ const SYSTEMTIME *lpDate, _In_<br>lpFormat, _Out_opt_ LPSTR lpDateStr, _In_                                       |
| wuitwwiwi | int    | GetDateFormatEx(_In_opt_ LPCWSTR lpLd<br>DWORD dwFlags, _In_opt_ const SYSTEM<br>_In_opt_ LPCWSTR lpFormat, _Out_opt_ L<br>lpDateStr, _In_ int cchDate, _In_opt_ LPCW |
| uiuitwwii | int    | GetDateFormatW(_In_ LCID Locale, _In_ D<br>dwFlags, _In_opt_ const SYSTEMTIME *lp<br>LPCWSTR lpFormat, _Out_opt_ LPWSTR l<br>int cchDate)                             |
| stti      | BOOL   | GetDefaultCommConfig(_In_ LPCTSTR lps<br>LPCOMMCONFIG lpCC, _Inout_ LPDWOI                                                                                            |
| atti      | BOOL   | GetDefaultCommConfigA(_In_ LPCSTR lps<br>LPCOMMCONFIG lpCC, _Inout_ LPDWOI                                                                                            |
| wtti      | BOOL   | GetDefaultCommConfigW(_In_ LPCWSTR<br>_Out_ LPCOMMCONFIG lpCC, _Inout_ LP<br>lpdwSize)                                                                                |
| titi      | BOOL   | GetDevicePowerState(_In_ HANDLE hDevi<br>*pfOn)                                                                                                                       |

|              |       |                                                                                                                                                                                                 |
|--------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sttti        | BOOL  | GetDiskFreeSpace(_In_ LPCTSTR lpRootPa<br>LPDWORD lpSectorsPerCluster, _Out_ LPD<br>lpBytesPerSector, _Out_ LPDWORD<br>lpNumberOfFreeClusters, _Out_ LPDWORD<br>lpTotalNumberOfClusters)        |
| attti        | BOOL  | GetDiskFreeSpaceA(_In_ LPCSTR lpRootPa<br>LPDWORD lpSectorsPerCluster, _Out_ LPD<br>lpBytesPerSector, _Out_ LPDWORD<br>lpNumberOfFreeClusters, _Out_ LPDWORD<br>lpTotalNumberOfClusters)        |
| stti         | BOOL  | GetDiskFreeSpaceEx(_In_opt_ LPCTSTR lp<br>_Out_opt_ PULARGE_INTEGER lpFreeByt<br>_Out_opt_ PULARGE_INTEGER lpTotalNu<br>_Out_opt_ PULARGE_INTEGER<br>lpTotalNumberOfFreeBytes)                  |
| atiti        | BOOL  | GetDiskFreeSpaceExA(_In_opt_ LPCSTR lp<br>_Out_opt_ PULARGE_INTEGER lpFreeByt<br>_Out_opt_ PULARGE_INTEGER lpTotalNu<br>_Out_opt_ PULARGE_INTEGER<br>lpTotalNumberOfFreeBytes)                  |
| wttti        | BOOL  | GetDiskFreeSpaceExW(_In_opt_ LPCWSTR<br>lpDirectoryName, _Out_opt_ PULARGE_IN<br>lpFreeBytesAvailable, _Out_opt_ PULARGE<br>lpTotalNumberOfBytes, _Out_opt_ PULARG<br>lpTotalNumberOfFreeBytes) |
| wttti        | BOOL  | GetDiskFreeSpaceW(_In_ LPCWSTR lpRoo<br>_Out_ LPDWORD lpSectorsPerCluster, _Ou<br>lpBytesPerSector, _Out_ LPDWORD<br>lpNumberOfFreeClusters, _Out_ LPDWORD<br>lpTotalNumberOfClusters)          |
| uisui        | DWORD | GetDllDirectory(_In_ DWORD nBufferLeng<br>LPCTSTR lpBuffer)                                                                                                                                     |
| uiaui        | DWORD | GetDllDirectoryA(_In_ DWORD nBufferLen<br>LPSTR lpBuffer)                                                                                                                                       |
| uiwui        | DWORD | GetDllDirectoryW(_In_ DWORD nBufferLen<br>LPWSTR lpBuffer)                                                                                                                                      |
| sui          | UINT  | GetDriveType(_In_opt_ LPCTSTR lpRootPa                                                                                                                                                          |
| au           | UINT  | GetDriveTypeA(_In_opt_ LPCSTR lpRootPa                                                                                                                                                          |
| wui          | UINT  | GetDriveTypeW(_In_opt_ LPCWSTR lpRoo                                                                                                                                                            |
| uiuitui6wwii | int   | GetDurationFormat(_In_ LCID Locale, _In_<br>dwFlags, _In_opt_ const SYSTEMTIME *lp<br>ULONGLONG ullDuration, _In_opt_ LPCW<br>_Out_opt_ LPWSTR lpDurationStr, _In_ int                          |
| wuitui6wwii  | int   | GetDurationFormatEx(_In_opt_ LPCWSTR<br>_In_ DWORD dwFlags, _In_opt_ const SYS<br>*lpDuration, _In_ ULONGLONG ullDuratio                                                                        |

|          |         |                                                                                                                         |
|----------|---------|-------------------------------------------------------------------------------------------------------------------------|
|          |         | LPCWSTR lpFormat, _Out_opt_ LPWSTR l<br>_In_ int cchDuration)                                                           |
| tui      | DWORD   | GetDynamicTimeZoneInformation(_Out_<br>PDYNAMIC_TIME_ZONE_INFORMATION<br>pTimeZoneInformation)                          |
| ui6      | DWORD64 | GetEnabledXStateFeatures(void)                                                                                          |
| s        | LPTCH   | GetEnvironmentString(void)                                                                                              |
| a        | LPCH    | GetEnvironmentStringsA(void)                                                                                            |
| w        | LPWCH   | GetEnvironmentStringsW(void)                                                                                            |
| ssuiui   | DWORD   | GetEnvironmentVariable(_In_opt_ LPCTSTR<br>_Out_opt_ LPTSTR lpBuffer, _In_ DWORD                                        |
| aauiui   | DWORD   | GetEnvironmentVariableA(_In_opt_ LPCSTR<br>_Out_opt_ LPSTR lpBuffer, _In_ DWORD r                                       |
| wwuiui   | DWORD   | GetEnvironmentVariableW(_In_opt_ LPCW<br>_Out_opt_ LPWSTR lpBuffer, _In_ DWOR                                           |
| ui       | UINT    | GetErrorMode(void)                                                                                                      |
| tui      | BOOL    | GetExitCodeProcess(_In_ HANDLE hProces<br>LPDWORD lpExitCode)                                                           |
| tui      | BOOL    | GetExitCodeThread(_In_ HANDLE hThread<br>LPDWORD lpExitCode)                                                            |
| ssi      | INT     | GetExpandedName(LPTSTR,LPTSTR)                                                                                          |
| aaui     | INT     | GetExpandedNameA(LPSTR,LPSTR)                                                                                           |
| wwui     | INT     | GetExpandedNameW(LPWSTR,LPWSTR)                                                                                         |
| sui      | DWORD   | GetFileAttributes(_In_ LPCTSTR lpFileNam                                                                                |
| auui     | DWORD   | GetFileAttributesA(_In_ LPCSTR lpFileNam                                                                                |
| suiui    | BOOL    | GetFileAttributesEx(_In_ LPCTSTR lpFileN<br>GET_FILEEX_INFO_LEVELS fInfoLevelI<br>LPVOID lpFileInformation)             |
| auuiui   | BOOL    | GetFileAttributesExA(_In_ LPCSTR lpFileN<br>GET_FILEEX_INFO_LEVELS fInfoLevelI<br>LPVOID lpFileInformation)             |
| wwuiui   | BOOL    | GetFileAttributesExW(_In_ LPCWSTR lpFi<br>GET_FILEEX_INFO_LEVELS fInfoLevelI<br>LPVOID lpFileInformation)               |
| suiuiui  | BOOL    | GetFileAttributesTransacted(_In_ LPCTSTR<br>_In_ GET_FILEEX_INFO_LEVELS fInfoL<br>LPVOID lpFileInformation, _In_ HANDLE |
| auuiuiui | BOOL    | GetFileAttributesTransactedA(_In_ LPCSTR<br>_In_ GET_FILEEX_INFO_LEVELS fInfoL<br>LPVOID lpFileInformation, _In_ HANDLE |
|          |         | GetFileAttributesTransactedW(_In_ LPCWS                                                                                 |

|          |       |                                                                                                                                                                                                           |
|----------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wuitti   | BOOL  | _In_ GET_FILEEX_INFO_LEVELS fInfoL<br>LPVOID lpFileInformation, _In_ HANDLE                                                                                                                               |
| wui      | DWORD | GetFileAttributesW(_In_ LPCWSTR lpFileN                                                                                                                                                                   |
| ttttti   | BOOL  | GetFileBandwidthReservation(_In_ HANDL<br>LPDWORD lpPeriodMilliseconds, _Out_ LP<br>lpBytesPerPeriod, _Out_ LPBOOL pDiscard<br>LPDWORD lpTransferSize, _Out_ LPDWOI<br>lpNumOutstandingRequests)          |
| tii      | BOOL  | GetFileInformationByHandle(_In_ HANDLE<br>LPBY_HANDLE_FILE_INFORMATION lp                                                                                                                                 |
| tituii   | BOOL  | GetFileInformationByHandleEx(_In_ HAND<br>FILE_INFO_BY_HANDLE_CLASS FileInf<br>_Out_ LPVOID lpFileInformation, _In_ DW<br>dwBufferSize)                                                                   |
| uiwtti   | BOOL  | GetFileMUIInfo(_In_ DWORD dwFlags, _In<br>pcwszFilePath, _Inout_opt_ PFILEMUIINFO<br>_Inout_ DWORD *pcbFileMUIInfo)                                                                                       |
| uiwtttti | BOOL  | GetFileMUIPath(_In_ DWORD dwFlags, _In<br>pcwszFilePath, _Inout_opt_ PWSTR pwszLa<br>PULONG pcchLanguage, _Out_opt_ PWSTI<br>pwszFileMUIPath, _Inout_ PULONG pcchF<br>_Inout_ PULONGLONG pululEnumerator) |
| ttui     | DWORD | GetFileSize(_In_ HANDLE hFile, _Out_opt<br>lpFileSizeHigh)                                                                                                                                                |
| tii      | BOOL  | GetFileSizeEx(_In_ HANDLE hFile, _Out_<br>PLARGE_INTEGER lpFileSize)                                                                                                                                      |
| tttti    | BOOL  | GetFileTime(_In_ HANDLE hFile, _Out_opt<br>lpCreationTime, _Out_opt_ LPFILETIME lp<br>_Out_opt_ LPFILETIME lpLastWriteTime)                                                                               |
| tui      | DWORD | GetFileType(_In_ HANDLE hFile)                                                                                                                                                                            |
| tsuiuiui | DWORD | GetFinalPathNameByHandle(_In_ HANDLE<br>LPTSTR lpszFilePath, _In_ DWORD cchFile<br>DWORD dwFlags)                                                                                                         |
| tauiuiui | DWORD | GetFinalPathNameByHandleA(_In_ HANDL<br>LPSTR lpszFilePath, _In_ DWORD cchFileF<br>DWORD dwFlags)                                                                                                         |
| twuiuiui | DWORD | GetFinalPathNameByHandleW(_In_ HAND<br>LPWSTR lpszFilePath, _In_ DWORD cchFi<br>DWORD dwFlags)                                                                                                            |
| sstuiui  | DWORD | GetFirmwareEnvironmentVariable(_In_ LPC<br>_In_ LPCTSTR lpGuid, _Out_ PVOID pBuffer<br>DWORD nSize)                                                                                                       |
| aatuiui  | DWORD | GetFirmwareEnvironmentVariableA(_In_ LP<br>_In_ LPCSTR lpGuid, _Out_ PVOID pBuffer<br>nSize)                                                                                                              |

|           |        |                                                                                                                                                             |
|-----------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwtuiu    | DWORD  | GetFirmwareEnvironmentVariableW(_In_ LPCTSTR lpName, _In_ LPCWSTR lpGuid, _Out_ PVOID lpBuffer, _In_ DWORD nSize)                                           |
| suistui   | DWORD  | GetFullPathName(_In_ LPCTSTR lpFileName, _Out_ LPCTSTR lpFullPath, _In_ LPCTSTR *lpFilePart)                                                                |
| auiatui   | DWORD  | GetFullPathNameA(_In_ LPCSTR lpFileName, _Out_ LPSTR lpFullPath, _In_ LPSTR *lpFilePart)                                                                    |
| suisttui  | DWORD  | GetFullPathNameTransacted(_In_ LPCTSTR lpFileName, _In_ DWORD nBufferLength, _Out_ LPCTSTR lpFullPath, _Out_ LPCTSTR *lpFilePart, _In_ HANDLE hTransaction) |
| auiattui  | DWORD  | GetFullPathNameTransactedA(_In_ LPCSTR lpFileName, _In_ DWORD nBufferLength, _Out_ LPSTR lpFullPath, _Out_ LPSTR *lpFilePart, _In_ HANDLE hTransaction)     |
| wuiwtui   | DWORD  | GetFullPathNameTransactedW(_In_ LPCWSTR lpFileName, _In_ DWORD nBufferLength, _Out_ LPWSTR lpFullPath, _Out_ LPWSTR *lpFilePart, _In_ HANDLE hTransaction)  |
| wuiwtui   | DWORD  | GetFullPathNameW(_In_ LPCWSTR lpFileName, _Out_ LPWSTR lpFullPath, _In_ LPWSTR *lpFilePart)                                                                 |
| uiuisiuhi | int    | GetGeoInfo(_In_ GEOID Location, _In_ GEOID GeoType, _Out_opt_ LPCTSTR lpGeoData, _In_ LANGID LangId)                                                        |
| uiuiaiuhi | int    | GetGeoInfoA(_In_ GEOID Location, _In_ GEOID GeoType, _Out_opt_ LPSTR lpGeoData, _In_ LANGID LangId)                                                         |
| uiuwiuhi  | int    | GetGeoInfoW(_In_ GEOID Location, _In_ GEOID GeoType, _Out_opt_ LPWSTR lpGeoData, _In_ LANGID LangId)                                                        |
| tui       | BOOL   | GetHandleInformation(_In_ HANDLE hObject, _Out_ LPDWORD lpdwFlags)                                                                                          |
| t         | SIZE_T | GetLargePageMinimum(void)                                                                                                                                   |
| tui       | COORD  | GetLargestConsoleWindowSize(_In_ HANDLE hConsoleOutput)                                                                                                     |
| ui        | DWORD  | GetLastError(void)                                                                                                                                          |
| uiuisi    | int    | GetLocaleInfo(_In_ LCID Locale, _In_ LCTYPE LCType, _Out_opt_ LPCTSTR lpLCData, _In_ int cchData)                                                           |
| uiuiai    | int    | GetLocaleInfoA(_In_ LCID Locale, _In_ LCTYPE LCType, _Out_opt_ LPSTR lpLCData, _In_ int cchData)                                                            |
| wuiwii    | int    | GetLocaleInfoEx(_In_opt_ LPCWSTR lpLocaleName, _In_ LCTYPE LCType, _Out_opt_ LPWSTR lpLCData, _In_ int cchData)                                             |

|         |       |                                                                                                                                                                                 |
|---------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiwii | int   | GetLocaleInfoW(_In_ LCID Locale, _In_ LOCALE_NAME LocaleName, _Out_opt_ LPWSTR lpLCData, _In_ int cchData)                                                                      |
| ti      | VOID  | GetLocalTime(_Out_ LPSYSTEMTIME lpSystemTime)                                                                                                                                   |
| ui      | DWORD | GetLogicalDrives(void)                                                                                                                                                          |
| uisui   | DWORD | GetLogicalDriveStrings(_In_ DWORD nBufferLength, _Out_ LPTSTR lpBuffer)                                                                                                         |
| uiaui   | DWORD | GetLogicalDriveStringsA(_In_ DWORD nBufferLength, _Out_ LPSTR lpBuffer)                                                                                                         |
| uiwui   | DWORD | GetLogicalDriveStringsW(_In_ DWORD nBufferLength, _Out_ LPWSTR lpBuffer)                                                                                                        |
| tti     | BOOL  | GetLogicalProcessorInformation(_Out_ PSYSTEM_LOGICAL_PROCESSOR_INFORMATION lpBuffer, _Inout_ PDWORD ReturnLength)                                                               |
| uitti   | BOOL  | GetLogicalProcessorInformationEx(_In_ LOGICAL_PROCESSOR_RELATIONSHIP RelationshipType, _Out_opt_ PSYSTEM_LOGICAL_PROCESSOR_INFORMATION lpBuffer, _Inout_ PDWORD ReturnedLength) |
| ssuiui  | DWORD | GetLongPathName(_In_ LPCTSTR lpszShortPath, _Out_ LPTSTR lpszLongPath, _In_ DWORD cchBuffer)                                                                                    |
| aauiui  | DWORD | GetLongPathNameA(_In_ LPCSTR lpszShortPath, _Out_ LPSTR lpszLongPath, _In_ DWORD cchBuffer)                                                                                     |
| ssuitui | DWORD | GetLongPathNameTransacted(_In_ LPCTSTR lpszShortPath, _Out_ LPTSTR lpszLongPath, _In_ DWORD cchBuffer, _In_ HANDLE hTransaction)                                                |
| aauitui | DWORD | GetLongPathNameTransactedA(_In_ LPCSTR lpszShortPath, _Out_ LPSTR lpszLongPath, _In_ DWORD cchBuffer, _In_ HANDLE hTransaction)                                                 |
| wwuitui | DWORD | GetLongPathNameTransactedW(_In_ LPCWSTR lpszShortPath, _Out_ LPWSTR lpszLongPath, _In_ DWORD cchBuffer, _In_ HANDLE hTransaction)                                               |
| wwuiui  | DWORD | GetLongPathNameW(_In_ LPCWSTR lpszShortPath, _Out_ LPWSTR lpszLongPath, _In_ DWORD cchBuffer)                                                                                   |
| tttti   | BOOL  | GetMailslotInfo(_In_ HANDLE hMailslot, _Out_ LPDWORD lpMaxMessageSize, _Out_opt_ LPDWORD lpNextSize, _Out_opt_ LPDWORD lpMessageSize, _Out_opt_ LPDWORD lpReadTimeout)          |
| uhui    | DWORD | GetMaximumProcessorCount(_In_ WORD Group)                                                                                                                                       |
| uh      | WORD  | GetMaximumProcessorGroupCount(void)                                                                                                                                             |
| tsuiui  | DWORD | GetModuleFileName(_In_opt_ HMODULE hModule, _Out_ LPTSTR lpFilename, _In_ DWORD nSize)                                                                                          |
| tauiui  | DWORD | GetModuleFileNameA(_In_opt_ HMODULE hModule, _Out_ LPSTR lpFilename, _In_ DWORD nSize)                                                                                          |

|          |         |                                                                                                                                                                                                                                            |
|----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twuiui   | DWORD   | GetModuleFileNameW(_In_opt_ HMODULE<br>_Out_ LPWSTR lpFilename, _In_ DWORD                                                                                                                                                                 |
| st       | HMODULE | GetModuleHandle(_In_opt_ LPCTSTR lpMo                                                                                                                                                                                                      |
| at       | HMODULE | GetModuleHandleA(_In_opt_ LPCSTR lpM                                                                                                                                                                                                       |
| uisti    | BOOL    | GetModuleHandleEx(_In_ DWORD dwFlag<br>LPCTSTR lpModuleName, _Out_ HMODUL                                                                                                                                                                  |
| uiati    | BOOL    | GetModuleHandleExA(_In_ DWORD dwFla<br>LPCSTR lpModuleName, _Out_ HMODUL                                                                                                                                                                   |
| uiwti    | BOOL    | GetModuleHandleExW(_In_ DWORD dwFL<br>LPCWSTR lpModuleName, _Out_ HMODU                                                                                                                                                                    |
| wt       | HMODULE | GetModuleHandleW(_In_opt_ LPCWSTR lp                                                                                                                                                                                                       |
| tsuii    | BOOL    | GetNamedPipeClientComputerName(_In_ H<br>_Out_ LPTSTR ClientComputerName, _In_<br>ClientComputerNameLength)                                                                                                                                |
| tauii    | BOOL    | GetNamedPipeClientComputerNameA(_In_<br>_Out_ LPSTR ClientComputerName, _In_ U<br>ClientComputerNameLength)                                                                                                                                |
| twuii    | BOOL    | GetNamedPipeClientComputerNameW(_In_<br>_Out_ LPWSTR ClientComputerName, _In_<br>ClientComputerNameLength)                                                                                                                                 |
| tii      | BOOL    | GetNamedPipeClientProcessId(_In_ HANDL<br>PULONG ClientProcessId)                                                                                                                                                                          |
| tii      | BOOL    | GetNamedPipeClientSessionId(_In_ HANDL<br>PULONG ClientSessionId)                                                                                                                                                                          |
| ttttsuii | BOOL    | GetNamedPipeHandleState(_In_ HANDLE h<br>_Out_opt_ LPDWORD lpState, _Out_opt_ L<br>lpCurInstances, _Out_opt_ LPDWORD<br>lpMaxCollectionCount, _Out_opt_ LPDWOI<br>lpCollectDataTimeout, _Out_opt_ LPTSTR l<br>_In_ DWORD nMaxUserNameSize) |
| ttttauui | BOOL    | GetNamedPipeHandleStateA(_In_ HANDLE<br>_Out_opt_ LPDWORD lpState, _Out_opt_ L<br>lpCurInstances, _Out_opt_ LPDWORD<br>lpMaxCollectionCount, _Out_opt_ LPDWOI<br>lpCollectDataTimeout, _Out_opt_ LPSTR lp<br>DWORD nMaxUserNameSize)       |
| ttttwuii | BOOL    | GetNamedPipeHandleStateW(_In_ HANDL<br>_Out_opt_ LPDWORD lpState, _Out_opt_ L<br>lpCurInstances, _Out_opt_ LPDWORD<br>lpMaxCollectionCount, _Out_opt_ LPDWOI<br>lpCollectDataTimeout, _Out_opt_ LPWSTR<br>_In_ DWORD nMaxUserNameSize)     |
| tttiti   | BOOL    | GetNamedPipeInfo(_In_ HANDLE hNamed<br>LPDWORD lpFlags, _Out_opt_ LPDWORD                                                                                                                                                                  |

|           |      |                                                                                                                                                              |
|-----------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |      | lpOutBufferSize, _Out_opt_ LPDWORD lpIn<br>_Out_opt_ LPDWORD lpMaxInstances)                                                                                 |
| tii       | BOOL | GetNamedPipeServerProcessId(_In_ HANDLE<br>PULONG ServerProcessId)                                                                                           |
| tii       | BOOL | GetNamedPipeServerSessionId(_In_ HANDLE<br>PULONG ServerSessionId)                                                                                           |
| ti        | VOID | GetNativeSystemInfo(_Out_ LPSYSTEM_INFO<br>lpSystemInfo)                                                                                                     |
| uiiiti    | BOOL | GetNLSVersion(_In_ NLS_FUNCTION Fun<br>LCID Locale, _Inout_ LPNLSVERSIONINFORM<br>lpVersionInformation)                                                      |
| uiwiti    | BOOL | GetNLSVersionEx(_In_ NLS_FUNCTION f<br>_In_opt_ LPCWSTR lpLocaleName, _Inout_<br>LPNLSVERSIONINFOEX lpVersionInforma                                         |
| ucti      | BOOL | GetNumaAvailableMemoryNode(_In_ UCHAR<br>PULONGLONG AvailableBytes)                                                                                          |
| uhti      | BOOL | GetNumaAvailableMemoryNodeEx(_In_ US<br>_Out_ PULONGLONG AvailableBytes)                                                                                     |
| ti        | BOOL | GetNumaHighestNodeNumber(_Out_ PULO<br>HighestNodeNumber)                                                                                                    |
| tii       | BOOL | GetNumaNodeNumberFromHandle(_In_ HA<br>_Out_ PUSHORT NodeNumber)                                                                                             |
| ucti      | BOOL | GetNumaNodeProcessorMask(_In_ UCHAR<br>PULONGLONG ProcessorMask)                                                                                             |
| uhti      | BOOL | GetNumaNodeProcessorMaskEx(_In_ USHO<br>_Out_ PGROUP_AFFINITY ProcessorMask                                                                                  |
| ucti      | BOOL | GetNumaProcessorNode(_In_ UCHAR Proc<br>PCHAR NodeNumber)                                                                                                    |
| tii       | BOOL | GetNumaProcessorNodeEx(_In_ PPROCESS<br>Processor, _Out_ PUSHORT NodeNumber)                                                                                 |
| uiti      | BOOL | GetNumaProximityNode(_In_ ULONG Prox<br>PCHAR NodeNumber)                                                                                                    |
| uiti      | BOOL | GetNumaProximityNodeEx(_In_ ULONG P<br>_Out_ PUSHORT NodeNumber)                                                                                             |
| uiiustsii | int  | GetNumberFormat(_In_ LCID Locale, _In_<br>dwFlags, _In_ LPCTSTR lpValue, _In_opt_<br>NUMBERFMT *lpFormat, _Out_opt_ LPTS<br>lpNumberStr, _In_ int cchNumber) |
| uiiataii  | int  | GetNumberFormatA(_In_ LCID Locale, _In_<br>dwFlags, _In_ LPCSTR lpValue, _In_opt_ co<br>NUMBERFMT *lpFormat, _Out_opt_ LPST<br>_In_ int cchNumber)           |
|           |      | GetNumberFormatEx(_In_opt_ LPCWSTR l                                                                                                                         |

|           |       |                                                                                                                                                              |
|-----------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wuiwtwii  | int   | _In_ DWORD dwFlags, _In_ LPCWSTR lpValue, const NUMBERFMT *lpFormat, _Out_opt_ LPWSTR lpNumberStr, _In_ int cchNumber)                                       |
| uiuiwtwii | int   | GetNumberFormatW(_In_ LCID Locale, _In_ DWORD dwFlags, _In_ LPCWSTR lpValue, _In_opt_ NUMBERFMT *lpFormat, _Out_opt_ LPWSTR lpNumberStr, _In_ int cchNumber) |
| tii       | BOOL  | GetNumberOfConsoleInputEvents(_In_ HANDLE hConsoleInput, _Out_ LPDWORD lpNumberOfConsoleInputEvents)                                                         |
| ti        | BOOL  | GetNumberOfConsoleMouseButtons(_Out_ LPDWORD lpNumberOfMouseButtons)                                                                                         |
| ui        | UINT  | GetOEMCP(void)                                                                                                                                               |
| ui        | UINT  | GetOEMCP(void)                                                                                                                                               |
| tttii     | BOOL  | GetOverlappedResult(_In_ HANDLE hFile, _In_ LPOVERLAPPED lpOverlapped, _Out_ LPDWORD lpNumberOfBytesTransferred, _In_ BOOL bWait)                            |
| ti        | BOOL  | GetPhysicallyInstalledSystemMemory(_Out_ PULONGLONG TotalMemoryInKilobytes)                                                                                  |
| tui       | DWORD | GetPriorityClass(_In_ HANDLE hProcess)                                                                                                                       |
| ssisui    | UINT  | GetPrivateProfileInt(_In_ LPCTSTR lpAppName, LPCTSTR lpKeyName, _In_ INT nDefault, _In_ LPCTSTR lpFileName)                                                  |
| aaiaui    | UINT  | GetPrivateProfileIntA(_In_ LPCSTR lpAppName, LPCSTR lpKeyName, _In_ INT nDefault, _In_ LPCSTR lpFileName)                                                    |
| wwiwui    | UINT  | GetPrivateProfileIntW(_In_ LPCWSTR lpAppName, LPCWSTR lpKeyName, _In_ INT nDefault, _In_ LPCWSTR lpFileName)                                                 |
| ssuisui   | DWORD | GetPrivateProfileSection(_In_ LPCTSTR lpAppName, LPTSTR lpReturnedString, _In_ DWORD nSize, _In_ LPCTSTR lpFileName)                                         |
| aauiaui   | DWORD | GetPrivateProfileSectionA(_In_ LPCSTR lpAppName, LPSTR lpReturnedString, _In_ DWORD nSize, _In_ LPCSTR lpFileName)                                           |
| suisui    | DWORD | GetPrivateProfileSectionNames(_Out_ LPTSTR lpszReturnBuffer, _In_ DWORD nSize, _In_ LPCTSTR lpFileName)                                                      |
| auiaui    | DWORD | GetPrivateProfileSectionNamesA(_Out_ LPSTR lpszReturnBuffer, _In_ DWORD nSize, _In_ LPCSTR lpFileName)                                                       |
| wuiwui    | DWORD | GetPrivateProfileSectionNamesW(_Out_ LPWSTR lpszReturnBuffer, _In_ DWORD nSize, _In_ LPCWSTR lpFileName)                                                     |
|           |       |                                                                                                                                                              |

|           |        |                                                                                                                                                  |
|-----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| wwuiwui   | DWORD  | GetPrivateProfileSectionW(_In_ LPCWSTR<br>_Out_ LPWSTR lpReturnedString, _In_ DW<br>_In_ LPCWSTR lpFileName)                                     |
| ssssuisui | DWORD  | GetPrivateProfileString(_In_ LPCTSTR lpA<br>LPCTSTR lpKeyName, _In_ LPCTSTR lpD<br>LPTSTR lpReturnedString, _In_ DWORD n<br>LPCTSTR lpFileName)  |
| aaaauiaui | DWORD  | GetPrivateProfileStringA(_In_ LPCSTR lpA<br>LPCSTR lpKeyName, _In_ LPCSTR lpDefa<br>LPSTR lpReturnedString, _In_ DWORD nSi<br>LPCSTR lpFileName) |
| wwwuiwui  | DWORD  | GetPrivateProfileStringW(_In_ LPCWSTR l<br>LPCWSTR lpKeyName, _In_ LPCWSTR lp<br>LPWSTR lpReturnedString, _In_ DWORD r<br>LPCWSTR lpFileName)    |
| sstuisi   | BOOL   | GetPrivateProfileStruct(_In_ LPCTSTR lpsz<br>LPCTSTR lpszKey, _Out_ LPVOID lpStruct<br>uSizeStruct, _In_ LPCTSTR szFile)                         |
| aatuiiai  | BOOL   | GetPrivateProfileStructA(_In_ LPCSTR lpsz<br>LPCSTR lpszKey, _Out_ LPVOID lpStruct,<br>uSizeStruct, _In_ LPCSTR szFile)                          |
| wwtuiwi   | BOOL   | GetPrivateProfileStructW(_In_ LPCWSTR l<br>LPCWSTR lpszKey, _Out_ LPVOID lpStruc<br>uSizeStruct, _In_ LPCWSTR szFile)                            |
| tat       | PROC   | GetProcAddress(_In_ HMODULE hModule,<br>lpProcName)                                                                                              |
| ttti      | BOOL   | GetProcessAffinityMask(_In_ HANDLE hPr<br>PDWORD_PTR lpProcessAffinityMask, _O<br>PDWORD_PTR lpSystemAffinityMask)                               |
| ttti      | BOOL   | GetProcessDEPPolicy(_In_ HANDLE hProc<br>LPDWORD lpFlags, _Out_ PBOOL lpPerma                                                                    |
| ttti      | BOOL   | GetProcessGroupAffinity(_In_ HANDLE hP<br>PUSHORT GroupCount, _Out_ PUSHORT G                                                                    |
| tti       | BOOL   | GetProcessHandleCount(_In_ HANDLE hPr<br>PDWORD pdwHandleCount)                                                                                  |
| t         | HANDLE | GetProcessHeap(void)                                                                                                                             |
| uitui     | DWORD  | GetProcessHeaps(_In_ DWORD NumberOf<br>PHANDLE ProcessHeaps)                                                                                     |
| tui       | DWORD  | GetProcessId(_In_ HANDLE Process)                                                                                                                |
| tui       | DWORD  | GetProcessIdOfThread(_In_ HANDLE Thre                                                                                                            |
| tti       | BOOL   | GetProcessIoCounters(_In_ HANDLE hProc<br>PIO_COUNTERS lpIoCounters)                                                                             |
|           |        | GetProcessorSystemCycleTime(_In_ USHO                                                                                                            |

|             |       |                                                                                                                                                   |
|-------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| uhtti       | BOOL  | PSYSTEM_PROCESSOR_CYCLE_TIME_ Buffer, _Inout_ PDWORD ReturnedLength)                                                                              |
| uitwti      | BOOL  | GetProcessPreferredUILanguages(_In_ DWORD _Out_ PULONG pulNumLanguages, _Out_ c pwszLanguagesBuffer, _Inout_ PULONG pcchLanguagesBuffer)          |
| titi        | BOOL  | GetProcessPriorityBoost(_In_ HANDLE hPr PBOOL pDisablePriorityBoost)                                                                              |
| titi        | BOOL  | GetProcessShutdownParameters(_Out_ LPD lpdwLevel, _Out_ LPDWORD lpdwFlags)                                                                        |
| tttiti      | BOOL  | GetProcessTimes(_In_ HANDLE hProcess, _ LPFILETIME lpCreationTime, _Out_ LPFIL lpExitTime, _Out_ LPFILETIME lpKernelTi LPFILETIME lpUserTime)     |
| uiui        | DWORD | GetProcessVersion(_In_ DWORD ProcessId)                                                                                                           |
| titi        | BOOL  | GetProcessWorkingSetSize(_In_ HANDLE h PSIZE_T lpMinimumWorkingSetSize, _Out_ lpMaximumWorkingSetSize)                                            |
| tttiti      | BOOL  | GetProcessWorkingSetSizeEx(_In_ HANDL _Out_ PSIZE_T lpMinimumWorkingSetSize PSIZE_T lpMaximumWorkingSetSize, _Out_ Flags)                         |
| uiuiuiuiiti | BOOL  | GetProductInfo(_In_ DWORD dwOSMajorV DWORD dwOSMinorVersion, _In_ DWORL dwSpMajorVersion, _In_ DWORD dwSpMir _Out_ PDWORD pdwReturnedProductType) |
| ssiui       | UINT  | GetProfileInt(_In_ LPCTSTR lpAppName, _ lpKeyName, _In_ INT nDefault)                                                                             |
| aauii       | UINT  | GetProfileIntA(_In_ LPCSTR lpAppName, _ lpKeyName, _In_ INT nDefault)                                                                             |
| wwiui       | UINT  | GetProfileIntW(_In_ LPCWSTR lpAppNam LPCWSTR lpKeyName, _In_ INT nDefault)                                                                        |
| ssuiui      | DWORD | GetProfileSection(_In_ LPCTSTR lpAppNam LPTSTR lpReturnedString, _In_ DWORD nS                                                                    |
| aauiui      | DWORD | GetProfileSectionA(_In_ LPCSTR lpAppNam LPSTR lpReturnedString, _In_ DWORD nSi                                                                    |
| wwuiui      | DWORD | GetProfileSectionW(_In_ LPCWSTR lpAppL LPWSTR lpReturnedString, _In_ DWORD r                                                                      |
| ssssuiui    | DWORD | GetProfileString(_In_ LPCTSTR lpAppNam LPCTSTR lpKeyName, _In_ LPCTSTR lpD LPTSTR lpReturnedString, _In_ DWORD nS                                 |
| aaaauiui    | DWORD | GetProfileStringA(_In_ LPCSTR lpAppNam lpKeyName, _In_ LPCSTR lpDefault, _Out_ lpReturnedString, _In_ DWORD nSize)                                |

|          |        |                                                                                                                                                                                                          |
|----------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwwwuiui | DWORD  | GetProfileStringW(_In_ LPCWSTR lpAppN<br>LPCWSTR lpKeyName, _In_ LPCWSTR lp<br>LPWSTR lpReturnedString, _In_ DWORD n                                                                                     |
| tttuii   | BOOL   | GetQueuedCompletionStatus(_In_ HANDLE<br>CompletionPort, _Out_ LPDWORD lpNumb<br>_Out_ PULONG_PTR lpCompletionKey, _C<br>LPOVERLAPPED *lpOverlapped, _In_ DW<br>dwMilliseconds)                          |
| ttuituii | BOOL   | GetQueuedCompletionStatusEx(_In_ HAND<br>CompletionPort, _Out_ LPOVERLAPPED_<br>lpCompletionPortEntries, _In_ ULONG ulCo<br>PULONG ulNumEntriesRemoved, _In_ DW<br>dwMilliseconds, _In_ BOOL fAlertable) |
| ssuiui   | DWORD  | GetShortPathName(_In_ LPCTSTR lpszLon<br>LPTSTR lpszShortPath, _In_ DWORD cchB                                                                                                                           |
| aauiui   | DWORD  | GetShortPathNameA(_In_ LPCSTR lpszLon<br>LPSTR lpszShortPath, _In_ DWORD cchBu                                                                                                                           |
| wwuiui   | DWORD  | GetShortPathNameW(_In_ LPCWSTR lpszL<br>LPWSTR lpszShortPath, _In_ DWORD cchF                                                                                                                            |
| ti       | VOID   | GetStartupInfo(_Out_ LPSTARTUPINFO lp                                                                                                                                                                    |
| ti       | VOID   | GetStartupInfoA(_Out_ LPSTARTUPINFO I                                                                                                                                                                    |
| ti       | VOID   | GetStartupInfoW(_Out_ LPSTARTUPINFO                                                                                                                                                                      |
| uit      | HANDLE | GetStdHandle(_In_ DWORD nStdHandle)                                                                                                                                                                      |
| uiwiwii  | int    | GetStringScripts(_In_ DWORD dwFlags, _In<br>lpString, _In_ int cchString, _Out_opt_ LPW<br>_In_ int cchScripts)                                                                                          |
| uiuisiti | BOOL   | GetStringType(LCID Locale, DWORD<br>dwInfoType, LPCTSTR lpSrcStr, int cchSrc, L<br>lpCharType)                                                                                                           |
| uiuiaiti | BOOL   | GetStringTypeA(_In_ LCID Locale, _In_ DV<br>dwInfoType, _In_ LPCSTR lpSrcStr, _In_ in<br>LPWORD lpCharType)                                                                                              |
| uiuisiti | BOOL   | GetStringTypeEx(_In_ LCID Locale, _In_ D<br>dwInfoType, _In_ LPCTSTR lpSrcStr, _In_ i<br>LPWORD lpCharType)                                                                                              |
| uiuiaiti | BOOL   | GetStringTypeExA(_In_ LCID Locale, _In_<br>dwInfoType, _In_ LPCSTR lpSrcStr, _In_ in<br>LPWORD lpCharType)                                                                                               |
| uiuiwiti | BOOL   | GetStringTypeExW(_In_ LCID Locale, _In_<br>dwInfoType, _In_ LPCWSTR lpSrcStr, _In_<br>_Out_ LPWORD lpCharType)                                                                                           |
| uiwiti   | BOOL   | GetStringTypeW(_In_ DWORD dwInfoType<br>LPCWSTR lpSrcStr, _In_ int cchSrc, _Out_<br>lpCharType)                                                                                                          |

|           |                        |                                                                                                                                                  |
|-----------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| uh        | LANGID                 | GetSystemDefaultLangID(void)                                                                                                                     |
| uh        | LANGID                 | GetSystemDefaultLangID(void)                                                                                                                     |
| ui        | LCID                   | GetSystemDefaultLCID(void)                                                                                                                       |
| ui        | LCID                   | GetSystemDefaultLCID(void)                                                                                                                       |
| wii       | int                    | GetSystemDefaultLocaleName(_Out_ LPWSTR lpLocaleName, _In_ int cchLocaleName)                                                                    |
| uh        | LANGID                 | GetSystemDefaultUILanguage(void)                                                                                                                 |
| ui        | DEP_SYSTEM_POLICY_TYPE | GetSystemDEFPolicy(void)                                                                                                                         |
| suiui     | UINT                   | GetSystemDirectory(_Out_ LPTSTR lpBuffer, uSize)                                                                                                 |
| auui      | UINT                   | GetSystemDirectoryA(_Out_ LPSTR lpBuffer, uSize)                                                                                                 |
| wuiui     | UINT                   | GetSystemDirectoryW(_Out_ LPWSTR lpBuffer, uSize)                                                                                                |
| tti       | BOOL                   | GetSystemFileCacheSize(_Out_ PSIZE_T lpMinimumFileCacheSize, _Out_ PSIZE_T lpMaximumFileCacheSize, _Out_ PDWORD)                                 |
| uiuituiui | UINT                   | GetSystemFirmwareTable(_In_ DWORD FirmwareTableProviderSignature, _In_ DWORD FirmwareTableID, _Out_ PVOID pFirmwareTable, _In_ DWORD BufferSize) |
| ti        | VOID                   | GetSystemInfo(_Out_ LPSYSTEM_INFO lpSystemInfo)                                                                                                  |
| ti        | BOOL                   | GetSystemPowerStatus(_Out_ LPSYSTEM_POWER_STATUS lpSystemPowerStatus)                                                                            |
| uitwti    | BOOL                   | GetSystemPreferredUILanguages(_In_ DWORD dwNumLanguages, _Out_ cpwpszLanguagesBuffer, _Inout_ PULONG pcchLanguagesBuffer)                        |
| tti       | BOOL                   | GetSystemRegistryQuota(_Out_opt_ PDWORD pdwQuotaAllowed, _Out_opt_ PDWORD pdwQuotaUsed)                                                          |
| ti        | VOID                   | GetSystemTime(_Out_ LPSYSTEMTIME lpSystemTime)                                                                                                   |
| tti       | BOOL                   | GetSystemTimeAdjustment(_Out_ PDWORD lpTimeAdjustment, _Out_ PDWORD lpTimeAdjustmentDisabled, _Out_ PBOOL lpTimeAdjustmentDisabled)              |
| ti        | VOID                   | GetSystemTimeAsFileTime(_Out_ LPFILETIME lpSystemTimeAsFileTime)                                                                                 |
| tti       | BOOL                   | GetSystemTimes(_Out_opt_ LPFILETIME lpSystemTime, _Out_opt_ LPFILETIME lpKernelTime, _Out_opt_ LPFILETIME lpUserTime)                            |
| suiui     | UINT                   | GetSystemWindowsDirectory(_Out_ LPTSTR lpBuffer, UINT uSize)                                                                                     |
|           |                        |                                                                                                                                                  |

|          |       |                                                                                                                                         |
|----------|-------|-----------------------------------------------------------------------------------------------------------------------------------------|
| auiai    | UINT  | GetSystemWindowsDirectoryA(_Out_ LPST<br>UINT uSize)                                                                                    |
| wuiui    | UINT  | GetSystemWindowsDirectoryW(_Out_ LPW<br>_In_ UINT uSize)                                                                                |
| suiui    | UINT  | GetSystemWow64Directory(_Out_ LPTSTR<br>UINT uSize)                                                                                     |
| auiai    | UINT  | GetSystemWow64DirectoryA(_Out_ LPSTR<br>UINT uSize)                                                                                     |
| wuiui    | UINT  | GetSystemWow64DirectoryW(_Out_ LPWS<br>_In_ UINT uSize)                                                                                 |
| tuittui  | DWORD | GetTapeParameters(_In_ HANDLE hDevice,<br>dwOperation, _Out_ LPDWORD lpdwSize, _<br>lpTapeInformation)                                  |
| tuitttui | DWORD | GetTapePosition(_In_ HANDLE hDevice, _I<br>dwPositionType, _Out_ LPDWORD lpdwPa<br>LPDWORD lpdwOffsetLow, _Out_ LPDW<br>lpdwOffsetHigh) |
| tui      | DWORD | GetTapeStatus(_In_ HANDLE hDevice)                                                                                                      |
| ssuisui  | UINT  | GetTempFileName(_In_ LPCTSTR lpPathNa<br>LPCTSTR lpPrefixString, _In_ UINT uUniq<br>LPTSTR lpTempFileName)                              |
| aauiiai  | UINT  | GetTempFileNameA(_In_ LPCSTR lpPathN<br>LPCSTR lpPrefixString, _In_ UINT uUniqu<br>lpTempFileName)                                      |
| wwuiwui  | UINT  | GetTempFileNameW(_In_ LPCWSTR lpPat<br>LPCWSTR lpPrefixString, _In_ UINT uUnic<br>LPWSTR lpTempFileName)                                |
| uisui    | DWORD | GetTempPath(_In_ DWORD nBufferLength,<br>lpBuffer)                                                                                      |
| uiaui    | DWORD | GetTempPathA(_In_ DWORD nBufferLengt<br>lpBuffer)                                                                                       |
| uiwui    | DWORD | GetTempPathW(_In_ DWORD nBufferLeng<br>LPWSTR lpBuffer)                                                                                 |
| tui      | BOOL  | GetThreadContext(_In_ HANDLE hThread,<br>LPCONTEXT lpContext)                                                                           |
| ui       | DWORD | GetThreadErrorMode(void)                                                                                                                |
| tui      | BOOL  | GetThreadGroupAffinity(_In_ HANDLE hT<br>PGROUP_AFFINITY GroupAffinity)                                                                 |
| tui      | DWORD | GetThreadId(_In_ HANDLE Thread)                                                                                                         |
| tui      | BOOL  | GetThreadIdealProcessorEx(_In_ HANDLE<br>PPROCESSOR_NUMBER lpIdealProcessor                                                             |
| tui      | BOOL  | GetThreadIOPendingFlag(_In_ HANDLE hT                                                                                                   |

|           |           |                                                                                                                                                                            |
|-----------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |           | PBOOL lpIOIsPending)                                                                                                                                                       |
| ui        | LCID      | GetThreadLocale(void)                                                                                                                                                      |
| uitwti    | BOOL      | GetThreadPreferredUILanguages(_In_ DWORD _Out_ PULONG pulNumLanguages, _Out_ PCWSTR pwszLanguagesBuffer, _Inout_ PULONG pcchLanguagesBuffer)                               |
| ti        | int       | GetThreadPriority(_In_ HANDLE hThread)                                                                                                                                     |
| tti       | BOOL      | GetThreadPriorityBoost(_In_ HANDLE hThread, PBOOL pDisablePriorityBoost)                                                                                                   |
| tuiti     | BOOL      | GetThreadSelectorEntry(_In_ HANDLE hThread, DWORD dwSelector, _Out_ LPLDT_ENTRY lpSelectorEntry)                                                                           |
| tttti     | BOOL      | GetThreadTimes(_In_ HANDLE hThread, _Out_ LPFILETIME lpCreationTime, _Out_ LPFILETIME lpExitTime, _Out_ LPFILETIME lpKernelTime, _Out_ LPFILETIME lpUserTime)              |
| uh        | LANGID    | GetThreadUILanguage(void)                                                                                                                                                  |
| ui        | DWORD     | GetTickCount(void)                                                                                                                                                         |
| ui6       | ULONGLONG | GetTickCount64(void)                                                                                                                                                       |
| uiuitssii | int       | GetTimeFormat(_In_ LCID Locale, _In_ DWORD dwFlags, _In_opt_ const SYSTEMTIME *lpTime, _In_ LPCWSTR lpFormat, _Out_opt_ LPTSTR lpTimeStr, _In_ int cchTime)                |
| uiuitaaii | int       | GetTimeFormatA(_In_ LCID Locale, _In_ DWORD dwFlags, _In_opt_ const SYSTEMTIME *lpTime, _In_ LPCSTR lpFormat, _Out_opt_ LPSTR lpTimeStr, _In_ int cchTime)                 |
| wuitwwii  | int       | GetTimeFormatEx(_In_opt_ LPCWSTR lpFormat, _In_ DWORD dwFlags, _In_opt_ const SYSTEMTIME *lpTime, _In_opt_ LPCWSTR lpFormat, _Out_opt_ LPWSTR lpTimeStr, _In_ int cchTime) |
| uiuitwwii | int       | GetTimeFormatW(_In_ LCID Locale, _In_ DWORD dwFlags, _In_opt_ const SYSTEMTIME *lpTime, _In_ LPCWSTR lpFormat, _Out_opt_ LPWSTR lpTimeStr, _In_ int cchTime)               |
| tui       | DWORD     | GetTimeZoneInformation(_Out_ LPTIME_ZONE_INFORMATION lpTimeZoneInformation)                                                                                                |
| uhtti     | BOOL      | GetTimeZoneInformationForYear(_In_ USHORT Year, _In_opt_ PDYNAMIC_TIME_ZONE_INFORMATION pdtzi, _Out_ LPTIME_ZONE_INFORMATION lpTimeZoneInformation)                        |
| uiwwti    | BOOL      | GetUILanguageInfo(_In_ DWORD dwFlags, _Out_ PCZZWSTR pwmszLanguage, _Out_opt_ PCZZWSTR pwszFallbackLanguages, _Inout_opt_ PDWORD pcchFallbackLanguages, _Out_ PDWORD p...  |

|             |        |                                                                                                                                                                                                                                                                                                                |
|-------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uh          | LANGID | GetUserDefaultLangID(void)                                                                                                                                                                                                                                                                                     |
| uh          | LANGID | GetUserDefaultLangID(void)                                                                                                                                                                                                                                                                                     |
| ui          | LCID   | GetUserDefaultLCID(void)                                                                                                                                                                                                                                                                                       |
| ui          | LCID   | GetUserDefaultLCID(void)                                                                                                                                                                                                                                                                                       |
| wii         | int    | GetUserDefaultLocaleName(_Out_ LPWSTR<br>_In_ int cchLocaleName)                                                                                                                                                                                                                                               |
| uh          | LANGID | GetUserDefaultUILanguage(void)                                                                                                                                                                                                                                                                                 |
| uiui        | GEOID  | GetUserGeoID(_In_ GEOCLASS GeoClass)                                                                                                                                                                                                                                                                           |
| uitwti      | BOOL   | GetUserPreferredUILanguages(_In_ DWORD<br>_Out_ PULONG pulNumLanguages, _Out_ c<br>pwszLanguagesBuffer, _Inout_ PULONG<br>pcchLanguagesBuffer)                                                                                                                                                                 |
| ui          | DWORD  | GetVersion(void)                                                                                                                                                                                                                                                                                               |
| ti          | BOOL   | GetVersionEx(_Inout_ LPOSVERSIONINFO                                                                                                                                                                                                                                                                           |
| ti          | BOOL   | GetVersionExA(_Inout_ LPOSVERSIONIN<br>lpVersionInfo)                                                                                                                                                                                                                                                          |
| ti          | BOOL   | GetVersionExW(_Inout_ LPOSVERSIONIN<br>lpVersionInfo)                                                                                                                                                                                                                                                          |
| ssuitttsuii | BOOL   | GetVolumeInformation(_In_opt_ LPCTSTR<br>lpRootPathName, _Out_opt_ LPTSTR lpVol<br>_In_ DWORD nVolumeNameSize, _Out_opt<br>lpVolumeSerialNumber, _Out_opt_ LPDWO<br>lpMaximumComponentLength, _Out_opt_ L<br>lpFileSystemFlags, _Out_opt_ LPTSTR<br>lpFileSystemNameBuffer, _In_ DWORD<br>nFileSystemNameSize) |
| aauiittauii | BOOL   | GetVolumeInformationA(_In_opt_ LPCSTR<br>lpRootPathName, _Out_opt_ LPSTR lpVolu<br>_In_ DWORD nVolumeNameSize, _Out_opt<br>lpVolumeSerialNumber, _Out_opt_ LPDWO<br>lpMaximumComponentLength, _Out_opt_ L<br>lpFileSystemFlags, _Out_opt_ LPSTR<br>lpFileSystemNameBuffer, _In_ DWORD<br>nFileSystemNameSize)  |
| twuitttwuii | BOOL   | GetVolumeInformationByHandleW(_In_ HA<br>_Out_opt_ LPWSTR lpVolumeNameBuffer,<br>nVolumeNameSize, _Out_opt_ LPDWORD<br>lpVolumeSerialNumber, _Out_opt_ LPDWO<br>lpMaximumComponentLength, _Out_opt_ L<br>lpFileSystemFlags, _Out_opt_ LPWSTR<br>lpFileSystemNameBuffer, _In_ DWORD<br>nFileSystemNameSize)     |
|             |        | GetVolumeInformationW(_In_opt_ LPCWSTR<br>lpRootPathName, _Out_opt_ LPWSTR<br>lpVolumeNameBuffer, _In_ DWORD nVolu                                                                                                                                                                                             |

|             |      |                                                                                                                                                                                                     |
|-------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwuitttwuui | BOOL | GetVolumeSerialNumber(_Out_opt_ LPDWORD lpVolumeSerialNumber, LPDWORD lpMaximumComponentLength, LPDWORD lpFileSystemFlags, _Out_opt_ LPWSTR lpFileSystemNameBuffer, _In_ DWORD nFileSystemNameSize) |
| ssuui       | BOOL | GetVolumeNameForVolumeMountPoint(_In_ LPCSTR lpVolumeMountPoint, _Out_ LPTSTR lpszVolumeName, _In_ DWORD cchBufferLength)                                                                           |
| aauii       | BOOL | GetVolumeNameForVolumeMountPointA(_In_ LPCTSTR lpVolumeMountPoint, _Out_ LPSTR lpszVolumeName, _In_ DWORD cchBufferLength)                                                                          |
| wwuui       | BOOL | GetVolumeNameForVolumeMountPointW(_In_ LPWSTR lpVolumeMountPoint, _Out_ LPWSTR lpszVolumeName, _In_ DWORD cchBufferLength)                                                                          |
| ssuui       | BOOL | GetVolumePathName(_In_ LPCTSTR lpszFileVolumeName, LPTSTR lpszVolumePathName, _In_ DWORD cchBufferLength)                                                                                           |
| aauii       | BOOL | GetVolumePathNameA(_In_ LPCSTR lpszFileVolumeName, LPSTR lpszVolumePathName, _In_ DWORD cchBufferLength)                                                                                            |
| ssuui       | BOOL | GetVolumePathNamesForVolumeName(_In_ LPCSTR lpVolumeName, _Out_ LPTSTR lpszVolumePathNames, _In_ DWORD cchBufferLength, _Out_ PDWORD lpReturnLength)                                                |
| aauii       | BOOL | GetVolumePathNamesForVolumeNameA(_In_ LPCTSTR lpVolumeName, _Out_ LPSTR lpszVolumePathNames, _In_ DWORD cchBufferLength, _Out_ PDWORD lpReturnLength)                                               |
| ssuui       | BOOL | GetVolumePathNamesForVolumeNameW(_In_ LPWSTR lpVolumeName, _Out_ LPWSTR lpszVolumePathNames, _In_ DWORD cchBufferLength, _Out_ PDWORD lpReturnLength)                                               |
| ssuui       | BOOL | GetVolumePathNameW(_In_ LPCWSTR lpFileVolumeName, _Out_ LPWSTR lpszVolumePathName, _In_ DWORD cchBufferLength)                                                                                      |
| suiui       | UINT | GetWindowsDirectory(_Out_ LPTSTR lpBuffer, _In_ DWORD uSize)                                                                                                                                        |
| auuii       | UINT | GetWindowsDirectoryA(_Out_ LPSTR lpBuffer, _In_ DWORD uSize)                                                                                                                                        |
| wuiui       | UINT | GetWindowsDirectoryW(_Out_ LPWSTR lpBuffer, _In_ DWORD uSize)                                                                                                                                       |
| uittttui    | UINT | GetWriteWatch(_In_ DWORD dwFlags, _In_ LPVOID lpBaseAddress, _In_ SIZE_T dwRegionSize, _Out_ PULONG_PTR lpdwWriteWatch, _Inout_ PULONG_PTR lpdwGranularity)                                         |

|        |         |                                                                              |
|--------|---------|------------------------------------------------------------------------------|
| tti    | BOOL    | GetXStateFeaturesMask(_In_ PCONTEXT C<br>DWORD64 FeatureMask)                |
| suh    | ATOM    | GlobalAddAtom(_In_ LPCTSTR lpString)                                         |
| auh    | ATOM    | GlobalAddAtomA(_In_ LPCSTR lpString)                                         |
| wuh    | ATOM    | GlobalAddAtomW(_In_ LPCWSTR lpString)                                        |
| uitt   | HGLOBAL | GlobalAlloc(_In_ UINT uFlags, _In_ SIZE_T                                    |
| uit    | SIZE_T  | GlobalCompact(DWORD dwMinFree)                                               |
| uhuh   | ATOM    | GlobalDeleteAtom(_In_ ATOM nAtom)                                            |
| suh    | ATOM    | GlobalFindAtom(_In_ LPCTSTR lpString)                                        |
| auh    | ATOM    | GlobalFindAtomA(_In_ LPCSTR lpString)                                        |
| wuh    | ATOM    | GlobalFindAtomW(_In_ LPCWSTR lpString)                                       |
| ti     | VOID    | GlobalFix(HGLOBAL hMem)                                                      |
| tui    | UINT    | GlobalFlags(_In_ HGLOBAL hMem)                                               |
| tt     | HGLOBAL | GlobalFree(_In_ HGLOBAL hMem)                                                |
| uhsiui | UINT    | GlobalGetAtomName(_In_ ATOM nAtom, _<br>lpBuffer, _In_ int nSize)            |
| uhaiui | UINT    | GlobalGetAtomNameA(_In_ ATOM nAtom,<br>lpBuffer, _In_ int nSize)             |
| uhwiui | UINT    | GlobalGetAtomNameW(_In_ ATOM nAtom<br>LPWSTR lpBuffer, _In_ int nSize)       |
| tt     | HGLOBAL | GlobalHandle(_In_ LPVOID pMem)                                               |
| tt     | LPVOID  | GlobalLock(_In_ HGLOBAL hMem)                                                |
| ti     | VOID    | GlobalMemoryStatus(_Out_ LPMEMORYSTAT                                        |
| ti     | BOOL    | GlobalMemoryStatusEx(_Inout_ LPMEMOR<br>lpBuffer)                            |
| ttuit  | HGLOBAL | GlobalReAlloc(_In_ HGLOBAL hMem, _In<br>dwBytes, _In_ UINT uFlags)           |
| tt     | SIZE_T  | GlobalSize(_In_ HGLOBAL hMem)                                                |
| ti     | VOID    | GlobalUnfix(HGLOBAL hMem)                                                    |
| ti     | BOOL    | GlobalUnlock(_In_ HGLOBAL hMem)                                              |
| ti     | BOOL    | GlobalUnWire(HGLOBAL hMem)                                                   |
| tt     | LPVOID  | GlobalWire(HGLOBAL hMem)                                                     |
| tuiti  | BOOL    | Heap32First(_Inout_ LPHEAPENTRY32 lph<br>DWORD th32ProcessID, _In_ ULONG_PTR |
| tti    | BOOL    | Heap32ListFirst(_In_ HANDLE hSnapshot,<br>LPHEAPLIST32 lphl)                 |
| tti    | BOOL    | Heap32ListNext(_In_ HANDLE hSnapshot,<br>LPHEAPLIST32 lphl)                  |

|         |        |                                                                                                                                                                            |
|---------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti      | BOOL   | Heap32Next(_Out_ LPHEAPENTRY32 lphe                                                                                                                                        |
| tuitt   | LPVOID | HeapAlloc(_In_ HANDLE hHeap, _In_ DW<br>_In_ SIZE_T dwBytes)                                                                                                               |
| tuitt   | SIZE_T | HeapCompact(_In_ HANDLE hHeap, _In_ I<br>dwFlags)                                                                                                                          |
| uittt   | HANDLE | HeapCreate(_In_ DWORD flOptions, _In_ S<br>dwInitialSize, _In_ SIZE_T dwMaximumSiz                                                                                         |
| ti      | BOOL   | HeapDestroy(_In_ HANDLE hHeap)                                                                                                                                             |
| tuiti   | BOOL   | HeapFree(_In_ HANDLE hHeap, _In_ DWC<br>_In_ LPVOID lpMem)                                                                                                                 |
| ti      | BOOL   | HeapLock(_In_ HANDLE hHeap)                                                                                                                                                |
| tittti  | BOOL   | HeapQueryInformation(_In_opt_ HANDLE<br>_In_ HEAP_INFORMATION_CLASS<br>HeapInformationClass, _Out_ PVOID HeapI<br>SIZE_T HeapInformationLength, _Out_opt_<br>ReturnLength) |
| tuittt  | LPVOID | HeapReAlloc(_In_ HANDLE hHeap, _In_ D<br>dwFlags, _In_ LPVOID lpMem, _In_ SIZE_                                                                                            |
| titti   | BOOL   | HeapSetInformation(_In_opt_ HANDLE He<br>HEAP_INFORMATION_CLASS HeapInfor<br>_In_ PVOID HeapInformation, _In_ SIZE_T<br>HeapInformationLength)                             |
| tuitt   | SIZE_T | HeapSize(_In_ HANDLE hHeap, _In_ DWC<br>_In_ LPCVOID lpMem)                                                                                                                |
| ti      | BOOL   | HeapUnlock(_In_ HANDLE hHeap)                                                                                                                                              |
| tuiti   | BOOL   | HeapValidate(_In_ HANDLE hHeap, _In_ D<br>dwFlags, _In_opt_ LPCVOID lpMem)                                                                                                 |
| titi    | BOOL   | HeapWalk(_In_ HANDLE hHeap, _Inout_<br>LPPROCESS_HEAP_ENTRY lpEntry)                                                                                                       |
| uiwiwii | int    | IdnToAscii(_In_ DWORD dwFlags, _In_ LP<br>lpUnicodeCharStr, _In_ int cchUnicodeChar,<br>LPWSTR lpASCIIStr, _In_ int cchASCI                                                |
| uiwiwii | int    | IdnToNameprepUnicode(_In_ DWORD dwF<br>LPCWSTR lpUnicodeCharStr, _In_ int cchU<br>_Out_opt_ LPWSTR lpNameprepCharStr, _I<br>cchNameprepChar)                               |
| uiwiwii | int    | IdnToUnicode(_In_ DWORD dwFlags, _In_<br>lpASCIIStr, _In_ int cchASCIIStr, _O<br>LPWSTR lpUnicodeCharStr, _In_ int cchUn                                                   |
| uui     | BOOL   | InitAtomTable(_In_ DWORD nSize)                                                                                                                                            |
| ti      | VOID   | InitializeConditionVariable(_Out_<br>PCONDITION_VARIABLE ConditionVariab                                                                                                   |

|         |              |                                                                                                                                                                                  |
|---------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti  | BOOL         | <code>InitializeContext</code> (_Out_opt_ PVOID Buffer, ContextFlags, _Out_opt_ PCONTEXT *Context, _In_ DWORD ContextLength)                                                     |
| ti      | VOID         | <code>InitializeCriticalSection</code> (_Out_ LPCRITICAL_SECTION *lpCriticalSection)                                                                                             |
| tuii    | BOOL         | <code>InitializeCriticalSectionAndSpinCount</code> (_Out_ LPCRITICAL_SECTION lpCriticalSection, _In_ DWORD dwSpinCount)                                                          |
| tuiiii  | BOOL         | <code>InitializeCriticalSectionEx</code> (_Out_ LPCRITICAL_SECTION lpCriticalSection, _In_ DWORD dwSpinCount, _In_ DWORD dwFlags)                                                |
| tuiiuti | BOOL         | <code>InitializeProcThreadAttributeList</code> (_Out_opt_ LPPROC_THREAD_ATTRIBUTE_LIST lpAttributeList, _In_ DWORD dwAttributeCount, _Reserved_ dwFlags, _Inout_ PSIZE_T lpSize) |
| ti      | VOID         | <code>InitializeSLISTHead</code> (_Inout_ PSLIST_HEAD *pSListHead)                                                                                                               |
| ti      | VOID         | <code>InitializeSRWLock</code> (_Out_ PSRWLOCK *pSRWLock)                                                                                                                        |
| tuitti  | BOOL         | <code>InitOnceBeginInitialize</code> (_Inout_ LPINIT_ONCE *pInitOnce, _In_ DWORD dwFlags, _Out_ PBOOL *pPending, LPVOID *lpContext)                                              |
| tuiti   | BOOL         | <code>InitOnceComplete</code> (_Inout_ LPINIT_ONCE *pInitOnce, _In_ DWORD dwFlags, _In_opt_ LPVOID lpContext)                                                                    |
| ttti    | BOOL         | <code>InitOnceExecuteOnce</code> (_Inout_ PINIT_ONCE *pInitOnce, PINIT_ONCE_FN InitFn, _Inout_opt_ PVOID *pContext, _Out_opt_ LPVOID *Context)                                   |
| ti      | VOID         | <code>InitOnceInitialize</code> (_Out_ PINIT_ONCE *pInitOnce)                                                                                                                    |
| tuiuiui | LONG         | <code>InterlockedCompareExchange</code> (_Inout_ LONG volatile *Destination, _In_ LONG Exchange, _In_ LONG Comparand)                                                            |
| ti6i6i6 | LONGLONG     | <code>InterlockedCompareExchange64</code> (_Inout_ LONGLONG volatile *Destination, _In_ LONGLONG Exchange, _In_ LONGLONG Comparand)                                              |
| tui     | LONG         | <code>InterlockedDecrement</code> (_Inout_ LONG volatile *pValue)                                                                                                                |
| tuiui   | LONG         | <code>InterlockedExchange</code> (_Inout_ LONG volatile *pValue, _In_ LONG Value)                                                                                                |
| tuiui   | LONG         | <code>InterlockedExchangeAdd</code> (_Inout_ LONG volatile *pValue, _In_ LONG Value)                                                                                             |
| tt      | PSLIST_ENTRY | <code>InterlockedFlushSList</code> (_Inout_ PSLIST_HEAD *pSListHead)                                                                                                             |
| tui     | LONG         | <code>InterlockedIncrement</code> (_Inout_ LONG volatile *pValue)                                                                                                                |
| tt      | PSLIST_ENTRY | <code>InterlockedPopEntrySList</code> (_Inout_ PSLIST_HEAD *pSListHead)                                                                                                          |
| ttt     | PSLIST_ENTRY | <code>InterlockedPushEntrySList</code> (_Inout_ PSLIST_HEAD *pSListHead, _In_ PSLIST_ENTRY *pEntry)                                                                              |

|          |              |                                                                                                                                   |
|----------|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
|          |              | ListHead, _Inout_ PSLIST_ENTRY ListEntr                                                                                           |
| ttuit    | PSLIST_ENTRY | InterlockedPushListSList(_Inout_ PSLIST_F<br>ListHead, _Inout_ PSLIST_ENTRY List, _In<br>PSLIST_ENTRY ListEnd, _In_ ULONG Co      |
| ti       | BOOL         | IsBadCodePtr(_In_ FARPROC lpfn)                                                                                                   |
| tti      | BOOL         | IsBadHugeReadPtr(VOID* lp,UINT_PTR uc                                                                                             |
| tti      | BOOL         | IsBadHugeWritePtr(LPVOID lp,UINT_PTR                                                                                              |
| tti      | BOOL         | IsBadReadPtr(_In_ const VOID *lp, _In_ UI                                                                                         |
| sti      | BOOL         | IsBadStringPtr(_In_ LPCTSTR lpsz, _In_ UI<br>ucchMax)                                                                             |
| ati      | BOOL         | IsBadStringPtrA(_In_ LPCSTR lpsz, _In_ UI<br>ucchMax)                                                                             |
| wti      | BOOL         | IsBadStringPtrW(_In_ LPCWSTR lpsz, _In_<br>ucchMax)                                                                               |
| tti      | BOOL         | IsBadWritePtr(_In_ LPVOID lp, _In_ UINT_                                                                                          |
| uiuiuii  | BOOL         | IsCalendarLeapYear(_In_ CALID calId, _In_<br>_In_ UINT era)                                                                       |
| uci      | BOOL         | IsDBCSLeadByte(_In_ BYTE TestChar)                                                                                                |
| uiuci    | BOOL         | IsDBCSLeadByteEx(_In_ UINT CodePage, .<br>TestChar)                                                                               |
| i        | BOOL         | IsDebuggerPresent(void)                                                                                                           |
| uiuitwii | BOOL         | IsNLSDefinedString(_In_ NLS_FUNCTION<br>DWORD dwFlags, _In_ LPNLSVERSIONIN<br>lpVersionInformation, _In_ LPCWSTR lpStr<br>cchStr) |
| uiwii    | BOOL         | IsNormalizedString(_In_ NORM_FORM No<br>LPCWSTR lpString, _In_ int cwLength)                                                      |
| ttti     | BOOL         | IsProcessInJob(_In_ HANDLE ProcessHand<br>HANDLE JobHandle, _Out_ PBOOL Result                                                    |
| uii      | BOOL         | IsProcessorFeaturePresent(_In_ DWORD Pro                                                                                          |
| i        | BOOL         | IsSystemResumeAutomatic(void)                                                                                                     |
| i        | BOOL         | IsThreadAFiber(void)                                                                                                              |
| ti       | BOOL         | IsThreadPoolTimerSet(_Inout_ PTP_TIMER                                                                                            |
| uii      | BOOL         | IsValidCodePage(_In_ UINT CodePage)                                                                                               |
| uiuii    | BOOL         | IsValidLanguageGroup(_In_ LGRPID Langu<br>DWORD dwFlags)                                                                          |
| uiuii    | BOOL         | IsValidLocale(_In_ LCID Locale, _In_ DWC                                                                                          |
| wi       | BOOL         | IsValidLocaleName(_In_ LPCWSTR lpLoca                                                                                             |
| tti      | BOOL         | IsWow64Process(_In_ HANDLE hProcess, _                                                                                            |

|             |         |                                                                                                                                                                                                            |
|-------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |         | Wow64Process)                                                                                                                                                                                              |
| uiwiuii     | int     | LCIDToLocaleName(_In_ LCID Locale, _Out_ LPWSTR lpName, _In_ int cchName, _In_ DWORD dwFlags)                                                                                                              |
| uiuisisii   | int     | LCMapString(_In_ LCID Locale, _In_ DWORD dwMapFlags, _In_ LPCTSTR lpSrcStr, _In_ _Out_opt_ LPTSTR lpDestStr, _In_ int cchD                                                                                 |
| uiuiaiaii   | int     | LCMapStringA(_In_ LCID Locale, _In_ DWORD dwMapFlags, _In_ LPCSTR lpSrcStr, _In_ int _Out_opt_ LPSTR lpDestStr, _In_ int cchDe                                                                             |
| wuiwiwititi | int     | LCMapStringEx(_In_opt_ LPCWSTR lpLoc DWORD dwMapFlags, _In_ LPCWSTR lpS cchSrc, _Out_opt_ LPWSTR lpDestStr, _In_ _In_opt_ LPNLSVERSIONINFO lpVersionI _In_opt_ LPVOID lpReserved, _In_opt_ LP/ sortHandle) |
| uiuiwiwii   | int     | LCMapStringW(_In_ LCID Locale, _In_ DWORD dwMapFlags, _In_ LPCWSTR lpSrcStr, _In_ _Out_opt_ LPWSTR lpDestStr, _In_ int cchI                                                                                |
| ti          | VOID    | LeaveCriticalSection(_Inout_ LPCRITICAL_SECTION lpCriticalSection)                                                                                                                                         |
| titi        | VOID    | LeaveCriticalSectionWhenCallbackReturns(PTP_CALLBACK_INSTANCE pci, _Inout_ PCRITICAL_SECTION pcs)                                                                                                          |
| st          | HMODULE | LoadLibrary(_In_ LPCTSTR lpFileName)                                                                                                                                                                       |
| at          | HMODULE | LoadLibraryA(_In_ LPCSTR lpFileName)                                                                                                                                                                       |
| stuit       | HMODULE | LoadLibraryEx(_In_ LPCTSTR lpFileName, HANDLE hFile, _In_ DWORD dwFlags)                                                                                                                                   |
| atuit       | HMODULE | LoadLibraryExA(_In_ LPCSTR lpFileName, HANDLE hFile, _In_ DWORD dwFlags)                                                                                                                                   |
| wtuit       | HMODULE | LoadLibraryExW(_In_ LPCWSTR lpFileName, HANDLE hFile, _In_ DWORD dwFlags)                                                                                                                                  |
| wt          | HMODULE | LoadLibraryW(_In_ LPCWSTR lpFileName)                                                                                                                                                                      |
| atui        | DWORD   | LoadModule(_In_ LPCSTR lpModuleName, lpParameterBlock)                                                                                                                                                     |
| titi        | HGLOBAL | LoadResource(_In_opt_ HMODULE hModu hResInfo)                                                                                                                                                              |
| uiwwtuiti   | BOOL    | LoadStringByReference(_In_ DWORD Flag PCWSTR Language, _In_ PCWSTR SourceS _Out_opt_ PWSTR Buffer, _In_ ULONG cch _In_opt_ PCWSTR Directory, _Out_opt_ PU pchBufferOut)                                    |
| uitt        | HLOCAL  | LocalAlloc(_In_ UINT uFlags, _In_ SIZE_T                                                                                                                                                                   |

|             |        |                                                                                                                                                                                      |
|-------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uit         | SIZE_T | LocalCompact(UINT uMinFree)                                                                                                                                                          |
| wuiui       | LCID   | LocaleNameToLCID(_In_ LPCWSTR lpName, _Out_ DWORD dwFlags)                                                                                                                           |
| tii         | BOOL   | LocalFileTimeToFileTime(_In_ const FILETIME *lpLocalFileTime, _Out_ LPFILETIME lpFileTime)                                                                                           |
| tui         | UINT   | LocalFlag(_In_ HLOCAL hMem)                                                                                                                                                          |
| tt          | HLOCAL | LocalFree(_In_ HLOCAL hMem)                                                                                                                                                          |
| tt          | HLOCAL | LocalHandle(_In_ LPCVOID pMem)                                                                                                                                                       |
| tt          | LPVOID | LocalLock(_In_ HLOCAL hMem)                                                                                                                                                          |
| ttuit       | HLOCAL | LocalReAlloc(_In_ HLOCAL hMem, _In_ SIZE_T dwBytes, _In_ UINT uFlags)                                                                                                                |
| tuit        | SIZE_T | LocalShrink(HLOCAL hMem, UINT cbNewSize)                                                                                                                                             |
| tui         | UINT   | LocalSize(_In_ HLOCAL hMem)                                                                                                                                                          |
| ti          | BOOL   | LocalUnlock(_In_ HLOCAL hMem)                                                                                                                                                        |
| tuitt       | PVOID  | LocateXStateFeature(_In_ PCONTEXT Context, _In_ DWORD FeatureId, _Out_opt_ PDWORD lpFeatureId)                                                                                       |
| tuiuiuiiii  | BOOL   | LockFile(_In_ HANDLE hFile, _In_ DWORD dwFileOffsetLow, _In_ DWORD dwFileOffsetHigh, _In_ DWORD nNumberOfBytesToLockLow, _In_ DWORD nNumberOfBytesToLockHigh)                        |
| tuiuiuiiiti | BOOL   | LockFileEx(_In_ HANDLE hFile, _In_ DWORD dwFlags, _In_ DWORD dwReserved, _In_ DWORD nNumberOfBytesToLockLow, _In_ DWORD nNumberOfBytesToLockHigh, _Inout_ LPOVERLAPPED lpOverlapped) |
| tt          | LPVOID | LockResource(_In_ HGLOBAL hResData)                                                                                                                                                  |
| sss         | LPTSTR | lstrcat(_Inout_ LPTSTR lpString1, _In_ LPTSTR lpString2)                                                                                                                             |
| aaa         | LPSTR  | lstrcatA(_Inout_ LPSTR lpString1, _In_ LPSTR lpString2)                                                                                                                              |
| www         | LPWSTR | lstrcatW(_Inout_ LPWSTR lpString1, _In_ LPWSTR lpString2)                                                                                                                            |
| ssi         | int    | lstrcmp(_In_ LPCTSTR lpString1, _In_ LPCTSTR lpString2)                                                                                                                              |
| aa          | int    | lstrcmpA(_In_ LPCSTR lpString1, _In_ LPCSTR lpString2)                                                                                                                               |
| ssi         | int    | lstrcmpi(_In_ LPCTSTR lpString1, _In_ LPCTSTR lpString2)                                                                                                                             |
| aa          | int    | lstrcmpiA(_In_ LPCSTR lpString1, _In_ LPCSTR lpString2)                                                                                                                              |
| wwi         | int    | lstrcmpiW(_In_ LPCWSTR lpString1, _In_ LPCWSTR lpString2)                                                                                                                            |
| wwi         | int    | lstrcmpW(_In_ LPCWSTR lpString1, _In_ LPCWSTR lpString2)                                                                                                                             |
| sss         | LPTSTR | lstrcpy(_Out_ LPTSTR lpString1, _In_ LPTSTR lpString2)                                                                                                                               |

|              |        |                                                                                                                                                                                   |
|--------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aaa          | LPSTR  | lstrcpyA(_Out_ LPSTR lpString1, _In_ LPST                                                                                                                                         |
| ssis         | LPTSTR | lstrcpyt(_Out_ LPTSTR lpString1, _In_ LPC<br>lpString2, _In_ int iMaxLength)                                                                                                      |
| aaia         | LPSTR  | lstrcpynA(_Out_ LPSTR lpString1, _In_ LPC<br>_In_ int iMaxLength)                                                                                                                 |
| wwiw         | LPWSTR | lstrcpynW(_Out_ LPWSTR lpString1, _In_ L<br>lpString2, _In_ int iMaxLength)                                                                                                       |
| www          | LPWSTR | lstrcpwW(_Out_ LPWSTR lpString1, _In_ L<br>lpString2)                                                                                                                             |
| si           | int    | lstrlen(_In_ LPCTSTR lpString)                                                                                                                                                    |
| ai           | int    | lstrlenA(_In_ LPCSTR lpString)                                                                                                                                                    |
| wi           | int    | lstrlenW(_In_ LPCWSTR lpString)                                                                                                                                                   |
| ii           | VOID   | LZClose(INT)                                                                                                                                                                      |
| iiui         | LONG   | LZCopy(INT,INT)                                                                                                                                                                   |
| i            | VOID   | LZDown(VOID)                                                                                                                                                                      |
| ii           | INT    | LZInit(INT)                                                                                                                                                                       |
| stuhi        | INT    | LZOpenFile(LPTSTR,LPOFSTRUCT,WOR                                                                                                                                                  |
| atuhi        | INT    | LZOpenFileA(LPSTR,LPOFSTRUCT,WOR                                                                                                                                                  |
| wtuhi        | INT    | LZOpenFileW(LPWSTR,LPOFSTRUCT,WO                                                                                                                                                  |
| iaii         | INT    | LZRead(INT,LPSTR,INT)                                                                                                                                                             |
| iuiiui       | LONG   | LZSeek(INT,LONG,INT)                                                                                                                                                              |
| i            | INT    | LZStart(VOID)                                                                                                                                                                     |
| ttti         | BOOL   | MapUserPhysicalPages(_In_ PVOID lpAddr<br>ULONG_PTR NumberOfPages, _In_ PULO<br>UserPfnArray)                                                                                     |
| ttti         | BOOL   | MapUserPhysicalPagesScatter(_In_ PVOID<br>*VirtualAddresses, _In_ ULONG_PTR Num<br>_In_ PULONG_PTR PageArray)                                                                     |
| tuiuiuitt    | LPVOID | MapViewOfFile(_In_ HANDLE hFileMappi<br>DWORD dwDesiredAccess, _In_ DWORD<br>dwFileOffsetHigh, _In_ DWORD dwFileOff<br>SIZE_T dwNumberOfBytesToMap)                               |
| tuiuiuittt   | LPVOID | MapViewOfFileEx(_In_ HANDLE hFileMap<br>_In_ DWORD dwDesiredAccess, _In_ DWC<br>dwFileOffsetHigh, _In_ DWORD dwFileOff<br>SIZE_T dwNumberOfBytesToMap, _In_opt_<br>lpBaseAddress) |
| tuiuiuittuit | LPVOID | MapViewOfFileExNuma(_In_ HANDLE<br>hFileMappingObject, _In_ DWORD dwDesi<br>DWORD dwFileOffsetHigh, _In_ DWORD<br>dwFileOffsetLow, _In_ SIZE_T dwNumberC                          |

|          |      |                                                                                                                                                                                                               |
|----------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |      | <code>_In_opt_ LPVOID lpBaseAddress, _In_ DWORD ndPreferred)</code>                                                                                                                                           |
| tti      | BOOL | <code>Module32First(_In_ HANDLE hSnapshot, _In_ LPMODULEENTRY32 lpme)</code>                                                                                                                                  |
| tti      | BOOL | <code>Module32FirstW(_In_ HANDLE hSnapshot, _In_ LPMODULEENTRY32 lpme)</code>                                                                                                                                 |
| tti      | BOOL | <code>Module32Next(_In_ HANDLE hSnapshot, _In_ LPMODULEENTRY32 lpme)</code>                                                                                                                                   |
| tti      | BOOL | <code>Module32NextW(_In_ HANDLE hSnapshot, _In_ LPMODULEENTRY32 lpme)</code>                                                                                                                                  |
| ssi      | BOOL | <code>MoveFile(_In_ LPCTSTR lpExistingFileName, _In_ LPCTSTR lpNewFileName)</code>                                                                                                                            |
| aa       | BOOL | <code>MoveFileA(_In_ LPCSTR lpExistingFileName, _In_ LPCSTR lpNewFileName)</code>                                                                                                                             |
| ssui     | BOOL | <code>MoveFileEx(_In_ LPCTSTR lpExistingFileName, _In_ LPCTSTR lpNewFileName, _In_ DWORD dwFlags)</code>                                                                                                      |
| aaui     | BOOL | <code>MoveFileExA(_In_ LPCSTR lpExistingFileName, _In_ LPCSTR lpNewFileName, _In_ DWORD dwFlags)</code>                                                                                                       |
| wwui     | BOOL | <code>MoveFileExW(_In_ LPCWSTR lpExistingFileName, _In_opt_ LPCWSTR lpNewFileName, _In_ DWORD dwFlags)</code>                                                                                                 |
| ssttuiti | BOOL | <code>MoveFileTransacted(_In_ LPCTSTR lpExistingFileName, _In_opt_ LPCTSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_ DWORD dwFlags, _In_ HANDLE hTransaction)</code>  |
| aattuiti | BOOL | <code>MoveFileTransactedA(_In_ LPCSTR lpExistingFileName, _In_opt_ LPCSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_ DWORD dwFlags, _In_ HANDLE hTransaction)</code>   |
| wwttuiti | BOOL | <code>MoveFileTransactedW(_In_ LPCWSTR lpExistingFileName, _In_opt_ LPCWSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_ DWORD dwFlags, _In_ HANDLE hTransaction)</code> |
| wwi      | BOOL | <code>MoveFileW(_In_ LPCWSTR lpExistingFileName, _In_ LPCWSTR lpNewFileName)</code>                                                                                                                           |
| ssttui   | BOOL | <code>MoveFileWithProgress(_In_ LPCTSTR lpExistingFileName, _In_opt_ LPCTSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_ DWORD dwFlags)</code>                          |
| aattui   | BOOL | <code>MoveFileWithProgressA(_In_ LPCSTR lpExistingFileName, _In_opt_ LPCSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, LPVOID lpData, _In_ DWORD dwFlags)</code>                           |

|            |        |                                                                                                                                                                                              |
|------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |        | LPVOID lpData, _In_ DWORD dwFlags)                                                                                                                                                           |
| wwttuui    | BOOL   | MoveFileWithProgressW(_In_ LPCWSTR lpExistingFileName, _In_opt_ LPCWSTR lpNewFileName, _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine, _In_opt_ LPVOID lpData, _In_ DWORD dwFlags)            |
| iiii       | int    | MulDiv(int nNumber,int nNumerator,int nDenominator)                                                                                                                                          |
| uiuiaiwii  | int    | MultiByteToWideChar(_In_ UINT CodePage, _In_ DWORD dwFlags, _In_ LPCSTR lpMultiByteStr, _In_ int cbMultiByte, _Out_opt_ LPWSTR lpWideCharStr, _In_ int cchWideChar)                          |
| si         | BOOL   | NeedCurrentDirectoryForExePath(_In_ LPCSTR lpExeName)                                                                                                                                        |
| ai         | BOOL   | NeedCurrentDirectoryForExePathA(_In_ LPCSTR lpExeName)                                                                                                                                       |
| wi         | BOOL   | NeedCurrentDirectoryForExePathW(_In_ LPCWSTR lpExeName)                                                                                                                                      |
| uiwwuiti   | BOOL   | NotifyUILanguageChange(_In_ DWORD dwFlags, _In_ PCWSTR pcwstrNewLanguage, _In_opt_ PCWSTR pcwstrPreviousLanguage, _In_ DWORD dwFlags, _Out_opt_ PDWORD pdwStatusRtn)                         |
| uiist      | HANDLE | OpenEvent(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCTSTR lpName)                                                                                                         |
| uiiat      | HANDLE | OpenEventA(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCSTR lpName)                                                                                                         |
| uiiwt      | HANDLE | OpenEventW(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCWSTR lpName)                                                                                                        |
| atuit      | HFILE  | OpenFile(_In_ LPCSTR lpFileName, _Out_ LPREOPEN_BUFFER lpReOpenBuff, _In_ UINT uStyle)                                                                                                       |
| ttuiuituit | HANDLE | OpenFileById(_In_ HANDLE hFile, _In_ LPFILE_ID_DESCRIPTOR lpFileID, _In_ DWORD dwDesiredAccess, _In_ DWORD dwShareMode, _In_ LPSECURITY_ATTRIBUTES lpSecurityAttributes, _In_ DWORD dwFlags) |
| uiist      | HANDLE | OpenFileMapping(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCTSTR lpName)                                                                                                   |
| uiiat      | HANDLE | OpenFileMappingA(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCSTR lpName)                                                                                                   |
| uiiwt      | HANDLE | OpenFileMappingW(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCWSTR lpName)                                                                                                  |
| uiist      | HANDLE | OpenJobObject(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandles, _In_ LPCTSTR lpName)                                                                                                    |
| uiiat      | HANDLE | OpenJobObjectA(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandles, _In_ LPCSTR lpName)                                                                                                    |
|            |        | OpenJobObjectW(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandles, _In_ LPCWSTR lpName)                                                                                                   |

|        |        |                                                                                                                                 |
|--------|--------|---------------------------------------------------------------------------------------------------------------------------------|
| uiiwt  | HANDLE | BOOL bInheritHandles, _In_ LPCWSTR lpName)                                                                                      |
| uiist  | HANDLE | OpenMutex(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCTSTR lpName)                                                 |
| uiiat  | HANDLE | OpenMutexA(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCSTR lpName)                                                 |
| uiiwt  | HANDLE | OpenMutexW(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCWSTR lpName)                                                |
| tst    | HANDLE | OpenPrivateNamespace(_In_ LPVOID lpBoundaryDescriptor, _In_ LPCTSTR lpAlias)                                                    |
| tat    | HANDLE | OpenPrivateNamespaceA(_In_ LPVOID lpBoundaryDescriptor, _In_ LPCSTR lpAlias)                                                    |
| twt    | HANDLE | OpenPrivateNamespaceW(_In_ LPVOID lpBoundaryDescriptor, _In_ LPCWSTR lpAlias)                                                   |
| uiiuit | HANDLE | OpenProcess(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ DWORD dwProcessId)                                            |
| uiist  | HANDLE | OpenSemaphore(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCTSTR lpName)                                             |
| uiiat  | HANDLE | OpenSemaphoreA(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCSTR lpName)                                             |
| uiiwt  | HANDLE | OpenSemaphoreW(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCWSTR lpName)                                            |
| uiiuit | HANDLE | OpenThread(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ DWORD dwThreadId)                                              |
| uiist  | HANDLE | OpenWaitableTimer(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCTSTR lpTimerName)                                    |
| uiiat  | HANDLE | OpenWaitableTimerA(_In_ DWORD dwDesiredAccess, BOOL bInheritHandle, _In_ LPCSTR lpTimerName)                                    |
| uiiwt  | HANDLE | OpenWaitableTimerW(_In_ DWORD dwDesiredAccess, _In_ BOOL bInheritHandle, _In_ LPCWSTR lpTimerName)                              |
| si     | VOID   | OutputDebugString(_In_opt_ LPCTSTR lpString)                                                                                    |
| ai     | VOID   | OutputDebugStringA(_In_opt_ LPCSTR lpString)                                                                                    |
| wi     | VOID   | OutputDebugStringW(_In_opt_ LPCWSTR lpString)                                                                                   |
| ttuiti | BOOL   | PeekConsoleInput(_In_ HANDLE hConsole, PINPUT_RECORD lpBuffer, _In_ DWORD nNumberOfEventsToRead, LPDWORD lpNumberOfEventsRead)  |
| ttuiti | BOOL   | PeekConsoleInputA(_In_ HANDLE hConsole, PINPUT_RECORD lpBuffer, _In_ DWORD nNumberOfEventsToRead, LPDWORD lpNumberOfEventsRead) |
| ttuiti | BOOL   | PeekConsoleInputW(_In_ HANDLE hConsole, PINPUT_RECORD lpBuffer, _In_ DWORD nNumberOfEventsToRead, LPDWORD lpNumberOfEventsRead) |

|           |        |                                                                                                                                                                                                                       |
|-----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuittti  | BOOL   | PeekNamedPipe(_In_ HANDLE hNamedPipe, LPVOID lpBuffer, _In_ DWORD nBufferSize, LPDWORD lpBytesRead, _Out_opt_ LPDWORD lpTotalBytesAvail, _Out_opt_ LPDWORD lpBytesLeftThisMessage)                                    |
| tuitti    | BOOL   | PostQueuedCompletionStatus(_In_ HANDLE CompletionPort, _In_ DWORD dwNumberOfBytesTransferred, _In_ ULONG dwCompletionKey, _In_opt_ LPOVERLAPPED lpOverlapped)                                                         |
| tii       | BOOL   | PowerClearRequest(_In_ HANDLE PowerRequest, POWER_REQUEST_TYPE RequestType)                                                                                                                                           |
| tt        | HANDLE | PowerCreateRequest(_In_ PREASON_CON...                                                                                                                                                                                |
| tii       | BOOL   | PowerSetRequest(_In_ HANDLE PowerRequest, POWER_REQUEST_TYPE RequestType)                                                                                                                                             |
| tuiiui    | DWORD  | PrepareTape(_In_ HANDLE hDevice, _In_ DWORD dwOperation, _In_ BOOL bImmediate)                                                                                                                                        |
| tii       | BOOL   | Process32First(_In_ HANDLE hSnapshot, _In_ LPPROCESSENTRY32 lppe)                                                                                                                                                     |
| tii       | BOOL   | Process32FirstW(_In_ HANDLE hSnapshot, _In_ LPPROCESSENTRY32 lppe)                                                                                                                                                    |
| tii       | BOOL   | Process32Next(_In_ HANDLE hSnapshot, _In_ LPPROCESSENTRY32 lppe)                                                                                                                                                      |
| tii       | BOOL   | Process32NextW(_In_ HANDLE hSnapshot, _In_ LPPROCESSENTRY32 lppe)                                                                                                                                                     |
| uiti      | BOOL   | ProcessIdToSessionId(_In_ DWORD dwProcessId, _Out_ DWORD *pSessionId)                                                                                                                                                 |
| ti        | BOOL   | PulseEvent(_In_ HANDLE hEvent)                                                                                                                                                                                        |
| tuii      | BOOL   | PurgeComm(_In_ HANDLE hFile, _In_ DWORD...                                                                                                                                                                            |
| uitwttti  | BOOL   | QueryActCtxSettingsW(_In_opt_ DWORD dwFlags, _In_opt_ HANDLE hActCtx, _In_opt_ PCWSTR settingsNameSpace, _In_ PCWSTR settingName, _Out_ PWSTR pvBuffer, _In_ SIZE_T dwBuffer, _Out_opt_ SIZE_T *pdwWrittenOrRequired) |
| uittuitti | BOOL   | QueryActCtxW(_In_ DWORD dwFlags, _In_ HANDLE hActCtx, _In_opt_ PVOID pvSubInstance, _In_ ULONG ulInfoClass, _Out_ PVOID pvBuffer, _In_opt_ ULONG cbBuffer, _Out_opt_ SIZE_T *pcbWrittenOrRequired)                    |
| tuh       | USHORT | QueryDepthSList(_In_ PSLIST_HEADER ListHead)                                                                                                                                                                          |
| ssuiui    | DWORD  | QueryDosDevice(_In_opt_ LPCTSTR lpDeviceName, _Out_ LPTSTR lpTargetPath, _In_ DWORD ucchMaximum)                                                                                                                      |
| aauiui    | DWORD  | QueryDosDeviceA(_In_opt_ LPCSTR lpDeviceName, _Out_ LPSTR lpTargetPath, _In_ DWORD ucchMaximum)                                                                                                                       |

|          |       |                                                                                                                                                                                               |
|----------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwuiui   | DWORD | QueryDosDeviceW(_In_opt_ LPCWSTR lpPath, _Out_ LPWSTR lpTargetPath, _In_ DWORD dwFlags)                                                                                                       |
| tuisti   | BOOL  | QueryFullProcessImageName(_In_ HANDLE hProcess, _In_ DWORD dwFlags, _Out_ LPTSTR lpExeName, _Out_ PDWORD lpdwSize)                                                                            |
| tuiati   | BOOL  | QueryFullProcessImageNameA(_In_ HANDLE hProcess, _In_ DWORD dwFlags, _Out_ LPSTR lpExeName, _Out_ PDWORD lpdwSize)                                                                            |
| tuiwti   | BOOL  | QueryFullProcessImageNameW(_In_ HANDLE hProcess, _In_ DWORD dwFlags, _Out_ LPWSTR lpExeName, _Out_ PDWORD lpdwSize)                                                                           |
| titi     | BOOL  | QueryIdleProcessorCycleTime(_Inout_ PULONG BufferLength, _Out_ PULONG64 ProcessorIdleCycleTime)                                                                                               |
| uhtti    | BOOL  | QueryIdleProcessorCycleTimeEx(_In_ USHORT BufferLength, _Out_ PULONG ProcessorIdleCycleTime)                                                                                                  |
| tuituiti | BOOL  | QueryInformationJobObject(_In_opt_ HANDLE hJobObject, _In_ JOBOBJECTINFOCLASS JobObjectInfoClass, LPVOID lpJobObjectInfo, _In_ DWORD cbJobObjectInfoLength, _Out_opt_ LPDWORD lpReturnLength) |
| titi     | BOOL  | QueryMemoryResourceNotification(_In_ HANDLE ResourceNotificationHandle, _Out_ PBOOLEAN pBoolean)                                                                                              |
| ti       | BOOL  | QueryPerformanceCounter(_Out_ LARGE_INTEGER *lpPerformanceCount)                                                                                                                              |
| ti       | BOOL  | QueryPerformanceFrequency(_Out_ LARGE_INTEGER *lpFrequency)                                                                                                                                   |
| tuii     | BOOL  | QueryProcessAffinityUpdateMode(_In_ HANDLE ProcessHandle, _Out_opt_ DWORD lpdwFlags)                                                                                                          |
| titi     | BOOL  | QueryProcessCycleTime(_In_ HANDLE ProcessHandle, _Out_ PULONG64 CycleTime)                                                                                                                    |
| titi     | BOOL  | QueryThreadCycleTime(_In_ HANDLE ThreadHandle, _Out_ PULONG64 CycleTime)                                                                                                                      |
| titi     | BOOL  | QueryThreadpoolStackInformation(_In_ PTP_POOL Pool, _Out_ PTP_POOL_STACK_INFORMATION pStackInformation)                                                                                       |
| ttui     | DWORD | QueryThreadProfiling(_In_ HANDLE ThreadHandle, _Out_ PBOOLEAN Enabled)                                                                                                                        |
| ti       | BOOL  | QueryUnbiasedInterruptTime(_Out_ PULONG UnbiasedTime)                                                                                                                                         |
| tttui    | DWORD | QueueUserAPC(_In_ PAPCFUNC pfnAPC, _In_ HANDLE hThread, _In_ ULONG_PTR dwData)                                                                                                                |
| ttuii    | BOOL  | QueueUserWorkItem(_In_ LPTHREAD_START_ROUTINE Function, _In_opt_ PVOID Context, _In_ ULONG Flags)                                                                                             |

|           |      |                                                                                                                                                                                                            |
|-----------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiuiti  | VOID | <a href="#">RaiseException</a> (_In_ DWORD dwException, DWORD dwExceptionFlags, _In_ DWORD nNumberOfArguments, _In_ const ULONG_*lpArguments)                                                              |
| ttuii     | VOID | <a href="#">RaiseFailFastException</a> (_In_opt_ PEXCEPTION_RECORD pExceptionRecord, _In_opt_ PCONTEXT pContext, _In_ DWORD dwFlags)                                                                       |
| ttuiti    | BOOL | <a href="#">ReadConsole</a> (_In_ HANDLE hConsoleInput, LPVOID lpBuffer, _In_ DWORD nNumberOfCharsToRead, _Out_ LPDWORD lpNumberOfCharsRead, _In_opt_ DWORD dwFlags, _In_opt_ INPUT_RECORD pInputControl)  |
| ttuiti    | BOOL | <a href="#">ReadConsoleA</a> (_In_ HANDLE hConsoleInput, LPVOID lpBuffer, _In_ DWORD nNumberOfCharsToRead, _Out_ LPDWORD lpNumberOfCharsRead, _In_opt_ DWORD dwFlags, _In_opt_ INPUT_RECORD pInputControl) |
| ttuiti    | BOOL | <a href="#">ReadConsoleInput</a> (_In_ HANDLE hConsoleInput, _Inout_ PINPUT_RECORD lpBuffer, _In_ DWORD nNumberOfEventsToRead, _Out_ LPDWORD lpNumberOfEventsRead)                                         |
| ttuiti    | BOOL | <a href="#">ReadConsoleInputA</a> (_In_ HANDLE hConsoleInput, _Inout_ PINPUT_RECORD lpBuffer, _In_ DWORD nNumberOfEventsToRead, _Out_ LPDWORD lpNumberOfEventsRead)                                        |
| ttuiti    | BOOL | <a href="#">ReadConsoleInputW</a> (_In_ HANDLE hConsoleInput, _Inout_ PINPUT_RECORD lpBuffer, _In_ DWORD nNumberOfEventsToRead, _Out_ LPDWORD lpNumberOfEventsRead)                                        |
| ttuiuiti  | BOOL | <a href="#">ReadConsoleOutput</a> (_In_ HANDLE hConsoleOutput, _Out_ PCHAR_INFO lpBuffer, _In_ COORD dwBufferCoord, _Inout_ PSMALL_RECT lpReadRegion)                                                      |
| ttuiuiti  | BOOL | <a href="#">ReadConsoleOutputA</a> (_In_ HANDLE hConsoleOutput, _Out_ PCHAR_INFO lpBuffer, _In_ COORD dwBufferCoord, _Inout_ PSMALL_RECT lpReadRegion)                                                     |
| ttuiuiti  | BOOL | <a href="#">ReadConsoleOutputAttribute</a> (_In_ HANDLE hConsoleOutput, _Out_ LPWORD lpAttribute, _In_ DWORD nLength, _In_ COORD dwReadCoord, _Out_ LPDWORD lpNumberOfAttrsRead)                           |
| tsuiuiti  | BOOL | <a href="#">ReadConsoleOutputCharacter</a> (_In_ HANDLE hConsoleOutput, _Out_ LPTSTR lpCharacter, _In_ DWORD nLength, _In_ COORD dwReadCoord, _Out_ LPDWORD lpNumberOfCharsRead)                           |
| tauuiuiti | BOOL | <a href="#">ReadConsoleOutputCharacterA</a> (_In_ HANDLE hConsoleOutput, _Out_ LPSTR lpCharacter, _In_ DWORD nLength, _In_ COORD dwReadCoord, _Out_ LPDWORD lpNumberOfCharsRead)                           |
| twuiuiti  | BOOL | <a href="#">ReadConsoleOutputCharacterW</a> (_In_ HANDLE hConsoleOutput, _Out_ LPWSTR lpCharacter, _In_ DWORD nLength, _In_ COORD dwReadCoord, _Out_ LPDWORD lpNumberOfCharsRead)                          |

|            |         |                                                                                                                                                                                                                                                                 |
|------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |         | DWORD nLength, _In_ COORD dwReadCo<br>LPDWORD lpNumberOfCharsRead)                                                                                                                                                                                              |
| ttuiuiti   | BOOL    | ReadConsoleOutputW(_In_ HANDLE hCon<br>_Out_ PCHAR_INFO lpBuffer, _In_ COOR<br>_In_ COORD dwBufferCoord, _Inout_ PSM<br>lpReadRegion)                                                                                                                           |
| ttuitti    | BOOL    | ReadConsoleW(_In_ HANDLE hConsoleInp<br>LPVOID lpBuffer, _In_ DWORD nNumberC<br>_Out_ LPDWORD lpNumberOfCharsRead, _<br>LPVOID pInputControl)                                                                                                                   |
| ttuiiuttti | BOOL    | ReadDirectoryChangesW(_In_ HANDLE hD<br>LPVOID lpBuffer, _In_ DWORD nBufferLe<br>BOOL bWatchSubtree, _In_ DWORD dwNo<br>_Out_opt_ LPDWORD lpBytesReturned, _In<br>LPOVERLAPPED lpOverlapped, _In_opt_<br>LPOVERLAPPED_COMPLETION_ROUTIN<br>lpCompletionRoutine) |
| ttuitti    | BOOL    | ReadFile(_In_ HANDLE hFile, _Out_ LPVC<br>_In_ DWORD nNumberOfBytesToRead, _O<br>LPDWORD lpNumberOfBytesRead, _Inout_<br>LPOVERLAPPED lpOverlapped)                                                                                                             |
| ttuitti    | BOOL    | ReadFileEx(_In_ HANDLE hFile, _Out_opt_<br>lpBuffer, _In_ DWORD nNumberOfBytesTo<br>LPOVERLAPPED lpOverlapped, _In_<br>LPOVERLAPPED_COMPLETION_ROUTIN<br>lpCompletionRoutine)                                                                                   |
| ttuitti    | BOOL    | ReadFileScatter(_In_ HANDLE hFile, _In_<br>FILE_SEGMENT_ELEMENT aSegmentArr<br>DWORD nNumberOfBytesToRead, _Reserv<br>lpReserved, _Inout_ LPOVERLAPPED lpOv                                                                                                     |
| tttiti     | BOOL    | ReadProcessMemory(_In_ HANDLE hProce<br>LPCVOID lpBaseAddress, _Out_ LPVOID l<br>SIZE_T nSize, _Out_ SIZE_T *lpNumberO                                                                                                                                          |
| tuitui     | DWORD   | ReadThreadProfilingData(_In_ HANDLE<br>PerformanceDataHandle, _In_ DWORD Flag<br>PPERFORMANCE_DATA PerformanceData                                                                                                                                              |
| ttuiiii    | HRESULT | RegisterApplicationRecoveryCallback(_In_<br>APPLICATION_RECOVERY_CALLBACK<br>pRecoveryCallback, _In_opt_ PVOID pvPar<br>DWORD dwPingInterval, _In_ DWORD dw                                                                                                     |
| wuii       | HRESULT | RegisterApplicationRestart(_In_opt_ PCWST<br>pwzCommandline, _In_ DWORD dwFlags)                                                                                                                                                                                |
| tttuiuui   | BOOL    | RegisterWaitForSingleObject(_Out_ PHAND<br>phNewWaitObject, _In_ HANDLE hObject,<br>WAITORTIMERCALLBACK Callback, _In_<br>Context, _In_ ULONG dwMilliseconds, _In_<br>dwFlags)                                                                                  |

|           |        |                                                                                                                                                                           |
|-----------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tttuiuitt | HANDLE | RegisterWaitForSingleObjectEx(HANDLE hObject, WAITORTIMERCALLBACK Callb Context, ULONG dwMilliseconds, ULONG d                                                            |
| ti        | VOID   | ReleaseActCtx(_In_ HANDLE hActCtx)                                                                                                                                        |
| ti        | BOOL   | ReleaseMutex(_In_ HANDLE hMutex)                                                                                                                                          |
| tii       | VOID   | ReleaseMutexWhenCallbackReturns(_Inout_ PTP_CALLBACK_INSTANCE pci, _In_ HA                                                                                                |
| tuiti     | BOOL   | ReleaseSemaphore(_In_ HANDLE hSemaph ReleaseCount, _Out_opt_ LPLONG lpPrevi                                                                                               |
| ttuui     | VOID   | ReleaseSemaphoreWhenCallbackReturns(_In PTP_CALLBACK_INSTANCE pci, _In_ HA _In_ DWORD crel)                                                                               |
| ti        | VOID   | ReleaseSRWLockExclusive(_Inout_ PSRWL SRWLock)                                                                                                                            |
| ti        | VOID   | ReleaseSRWLockShared(_Inout_ PSRWLOC                                                                                                                                      |
| si        | BOOL   | RemoveDirectory(_In_ LPCTSTR lpPathNa                                                                                                                                     |
| ai        | BOOL   | RemoveDirectoryA(_In_ LPCSTR lpPathNa                                                                                                                                     |
| sti       | BOOL   | RemoveDirectoryTransacted(_In_ LPCTSTR _In_ HANDLE hTransaction)                                                                                                          |
| ati       | BOOL   | RemoveDirectoryTransactedA(_In_ LPCSTR _In_ HANDLE hTransaction)                                                                                                          |
| wti       | BOOL   | RemoveDirectoryTransactedW(_In_ LPCWS _In_ HANDLE hTransaction)                                                                                                           |
| wi        | BOOL   | RemoveDirectoryW(_In_ LPCWSTR lpPath                                                                                                                                      |
| ti        | BOOL   | RemoveDllDirectory(_In_ DLL_DIRECTOR Cookie)                                                                                                                              |
| ti        | BOOL   | RemoveSecureMemoryCacheCallback(_In_ PSECURE_MEMORY_CACHE_CALLBA                                                                                                          |
| tui       | ULONG  | RemoveVectoredContinueHandler(_In_ PVC                                                                                                                                    |
| tui       | ULONG  | RemoveVectoredExceptionHandler(_In_ PVC                                                                                                                                   |
| tuiuiuitt | HANDLE | ReOpenFile(_In_ HANDLE hOriginalFile, _ dwDesiredAccess, _In_ DWORD dwShareM DWORD dwFlags)                                                                               |
| sssuiitti | BOOL   | ReplaceFile(_In_ LPCTSTR lpReplacedFileL LPCTSTR lpReplacementFileName, _In_opt_ lpBackupFileName, _In_ DWORD dwRepla _Reserved_ LPVOID lpExclude, _Reserved_ lpReserved) |
| aaauitti  | BOOL   | ReplaceFileA(_In_ LPCSTR lpReplacedFileL LPCSTR lpReplacementFileName, _In_opt_ lpBackupFileName, _In_ DWORD dwRepla                                                      |

|          |       |                                                                                                                                                                                                      |
|----------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |       | _Reserved_ LPVOID lpExclude, _Reserved_ lpReserved)                                                                                                                                                  |
| wwwuitti | BOOL  | ReplaceFileW(_In_ LPCWSTR lpReplacedF<br>LPCWSTR lpReplacementFileName, _In_ op<br>lpBackupFileName, _In_ DWORD dwRepla<br>_Reserved_ LPVOID lpExclude, _Reserved_ lpReserved)                       |
| ti       | BOOL  | RequestDeviceWakeup(HANDLE hDevice)                                                                                                                                                                  |
| ii       | BOOL  | RequestWakeupLatency(_In_ LATENCY_TI                                                                                                                                                                 |
| ti       | BOOL  | ResetEvent(_In_ HANDLE hEvent)                                                                                                                                                                       |
| ttui     | UINT  | ResetWriteWatch(_In_ LPVOID lpBaseAddr<br>SIZE_T dwRegionSize)                                                                                                                                       |
| wwii     | int   | ResolveLocaleName(_In_opt_ LPCWSTR<br>lpNameToResolve, _Out_opt_ LPWSTR lpL<br>int cchLocaleName)                                                                                                    |
| uii      | VOID  | RestoreLastError(DWORD dwErrCode)                                                                                                                                                                    |
| tui      | DWORD | ResumeThread(_In_ HANDLE hThread)                                                                                                                                                                    |
| ti       | VOID  | RtlCaptureContext(_Out_ PCONTEXT Cont                                                                                                                                                                |
| ttuci    | VOID  | RtlFillMemory(VOID UNALIGNED* Desti<br>Length, BYTE Fill)                                                                                                                                            |
| wuui     | BOOL  | RtlIsValidLocaleName(_In_ LPCWSTR Loc<br>ULONG Flags)                                                                                                                                                |
| ttti     | VOID  | RtlMoveMemory(VOID UNALIGNED* Des<br>UNALIGNED* Source, SIZE_T Length)                                                                                                                               |
| ttt      | PVOID | RtlPcToFileHeader(_In_ PVOID PcValue, _C<br>*BaseOfImage)                                                                                                                                            |
| tttti    | VOID  | RtlUnwind(_In_opt_ PVOID TargetFrame, _<br>TargetIp, _In_opt_ PEXCEPTION_RECORD<br>ExceptionRecord, _In_ PVOID ReturnValue)                                                                          |
| tti      | VOID  | RtlZeroMemory(VOID UNALIGNED* Dest<br>Length)                                                                                                                                                        |
| tttuiti  | BOOL  | ScrollConsoleScreenBuffer(_In_ HANDLE h<br>_In_ const SMALL_RECT *lpScrollRectang<br>const SMALL_RECT *lpClipRectangle, _In_<br>dwDestinationOrigin, _In_ const CHAR_INF                             |
| tttuiti  | BOOL  | ScrollConsoleScreenBufferA(_In_ HANDLE<br>hConsoleOutput, _In_ const SMALL_RECT<br>*lpScrollRectangle, _In_opt_ const SMALL_<br>*lpClipRectangle, _In_ COORD dwDestinati<br>const CHAR_INFO *lpFill) |
| tttuiti  | BOOL  | ScrollConsoleScreenBufferW(_In_ HANDLE<br>hConsoleOutput, _In_ const SMALL_RECT<br>*lpScrollRectangle, _In_opt_ const SMALL_                                                                         |

|           |       |                                                                                                                                                               |
|-----------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |       | *lpClipRectangle, _In_ COORD dwDestination, const CHAR_INFO *lpFill)                                                                                          |
| sssuistui | DWORD | SearchPath(_In_opt_ LPCTSTR lpPath, _In_ lpFileName, _In_opt_ LPCTSTR lpExtension, nBufferLength, _Out_ LPTSTR lpBuffer, _Out_ LPTSTR *lpFilePart)            |
| aauiatui  | DWORD | SearchPathA(_In_opt_ LPCSTR lpPath, _In_ lpFileName, _In_opt_ LPCSTR lpExtension, nBufferLength, _Out_ LPSTR lpBuffer, _Out_ *lpFilePart)                     |
| wwwuiwtui | DWORD | SearchPathW(_In_opt_ LPCWSTR lpPath, _In_ lpFileName, _In_opt_ LPCWSTR lpExtension, DWORD nBufferLength, _Out_ LPWSTR lpBuffer, _Out_opt_ LPWSTR *lpFilePart) |
| uiuiuisi  | BOOL  | SetCalendarInfo(_In_ LCID Locale, _In_ CA _In_ CALTYPE CalType, _In_ LPCTSTR lpC                                                                              |
| uiuiuai   | BOOL  | SetCalendarInfoA(_In_ LCID Locale, _In_ C _In_ CALTYPE CalType, _In_ LPCSTR lpC                                                                               |
| uiuiuiwi  | BOOL  | SetCalendarInfoW(_In_ LCID Locale, _In_ C _In_ CALTYPE CalType, _In_ LPCWSTR lp                                                                               |
| ti        | BOOL  | SetCommBreak(_In_ HANDLE hFile)                                                                                                                               |
| ttui      | BOOL  | SetCommConfig(_In_ HANDLE hCommDe LPCOMMCONFIG lpCC, _In_ DWORD dw                                                                                            |
| tui       | BOOL  | SetCommMask(_In_ HANDLE hFile, _In_ D dwEvtMask)                                                                                                              |
| tti       | BOOL  | SetCommState(_In_ HANDLE hFile, _In_ L                                                                                                                        |
| tti       | BOOL  | SetCommTimeouts(_In_ HANDLE hFile, _In_ LPCOMMTIMEOUTS lpCommTimeouts)                                                                                        |
| si        | BOOL  | SetComputerName(_In_ LPCTSTR lpComp                                                                                                                           |
| ai        | BOOL  | SetComputerNameA(_In_ LPCSTR lpComp                                                                                                                           |
| uisi      | BOOL  | SetComputerNameEx(_In_ COMPUTER_N NameType, _In_ LPCTSTR lpBuffer)                                                                                            |
| uiai      | BOOL  | SetComputerNameExA(_In_ COMPUTER_NAME_FORMAT NameType lpBuffer)                                                                                               |
| uiwi      | BOOL  | SetComputerNameExW(_In_ COMPUTER_NAME_FORMAT NameType LPCWSTR lpBuffer)                                                                                       |
| wi        | BOOL  | SetComputerNameW(_In_ LPCWSTR lpCor                                                                                                                           |
| ti        | BOOL  | SetConsoleActiveScreenBuffer(_In_ HANDL hConsoleOutput)                                                                                                       |
| uii       | BOOL  | SetConsoleCP(_In_ UINT wCodePageID)                                                                                                                           |

|       |       |                                                                                                                              |
|-------|-------|------------------------------------------------------------------------------------------------------------------------------|
| tii   | BOOL  | SetConsoleCtrlHandler(_In_opt_ PHANDLER_ROUTINE HandlerRoutine, _In_ BOOL Add)                                               |
| tii   | BOOL  | SetConsoleCursorInfo(_In_ HANDLE hConsoleOutput, _In_ const CONSOLE_CURSOR_INFO *lpConsoleCursorInfo)                        |
| tuii  | BOOL  | SetConsoleCursorPosition(_In_ HANDLE hConsoleOutput, _In_ COORD dwCursorPosition)                                            |
| tuiti | BOOL  | SetConsoleDisplayMode(_In_ HANDLE hConsoleOutput, _In_ DWORD dwFlags, _Out_opt_ PCOORD lpNewScreenBufferDimensions)          |
| ti    | BOOL  | SetConsoleHistoryInfo(_In_ PCONSOLE_HISTORY lpConsoleHistoryInfo)                                                            |
| tuii  | BOOL  | SetConsoleMode(_In_ HANDLE hConsoleOutput, _In_ DWORD dwMode)                                                                |
| uii   | BOOL  | SetConsoleOutputCP(_In_ UINT wCodePage)                                                                                      |
| tii   | BOOL  | SetConsoleScreenBufferInfoEx(_In_ HANDLE hConsoleOutput, _In_ PCONSOLE_SCREEN_BUFFER_INFOEX lpConsoleScreenBufferInfoEx)     |
| tuii  | BOOL  | SetConsoleScreenBufferSize(_In_ HANDLE hConsoleOutput, _In_ COORD dwSize)                                                    |
| tuhi  | BOOL  | SetConsoleTextAttribute(_In_ HANDLE hConsoleOutput, _In_ WORD wAttributes)                                                   |
| si    | BOOL  | SetConsoleTitle(_In_ LPCTSTR lpConsoleTitle)                                                                                 |
| ai    | BOOL  | SetConsoleTitleA(_In_ LPCSTR lpConsoleTitle)                                                                                 |
| wi    | BOOL  | SetConsoleTitleW(_In_ LPCWSTR lpConsoleTitle)                                                                                |
| titi  | BOOL  | SetConsoleWindowInfo(_In_ HANDLE hConsoleOutput, _In_ BOOL bAbsolute, _In_ const SMALL_RECT *lpConsoleWindow)                |
| tuiui | DWORD | SetCriticalSectionSpinCount(_Inout_ LPCRITICAL_SECTION lpCriticalSection, dwSpinCount)                                       |
| titi  | BOOL  | SetCurrentConsoleFontEx(_In_ HANDLE hConsoleOutput, _In_ BOOL bMaximumWindow, _In_ PCONSOLE_FONT_INFOEX lpConsoleFontInfoEx) |
| si    | BOOL  | SetCurrentDirectory(_In_ LPCTSTR lpPathName)                                                                                 |
| ai    | BOOL  | SetCurrentDirectoryA(_In_ LPCSTR lpPathName)                                                                                 |
| wi    | BOOL  | SetCurrentDirectoryW(_In_ LPCWSTR lpPathName)                                                                                |
| stuii | BOOL  | SetDefaultCommConfig(_In_ LPCTSTR lpszDeviceName, LPCOMMCONFIG lpCC, _In_ DWORD dwFlags)                                     |
| atuii | BOOL  | SetDefaultCommConfigA(_In_ LPCSTR lpszDeviceName, LPCOMMCONFIG lpCC, _In_ DWORD dwFlags)                                     |
|       |       |                                                                                                                              |

|       |      |                                                                                                     |
|-------|------|-----------------------------------------------------------------------------------------------------|
| wtuii | BOOL | SetDefaultCommConfigW(_In_ LPCWSTR LPCOMMCONFIG lpCC, _In_ DWORD dw                                 |
| uii   | BOOL | SetDefaultDllDirectories(_In_ DWORD Dire                                                            |
| si    | BOOL | SetDllDirectory(_In_opt_ LPCTSTR lpPathN                                                            |
| ai    | BOOL | SetDllDirectoryA(_In_opt_ LPCSTR lpPathN                                                            |
| wi    | BOOL | SetDllDirectoryW(_In_opt_ LPCWSTR lpPa                                                              |
| ti    | BOOL | SetDynamicTimeZoneInformation(_In_ cons<br>DYNAMIC_TIME_ZONE_INFORMATION<br>*lpTimeZoneInformation) |
| ti    | BOOL | SetEndOfFile(_In_ HANDLE hFile)                                                                     |
| ssi   | BOOL | SetEnvironmentVariable(_In_ LPCTSTR lpN<br>LPCTSTR lpValue)                                         |
| aaai  | BOOL | SetEnvironmentVariableA(_In_ LPCSTR lpN<br>LPCSTR lpValue)                                          |
| wwi   | BOOL | SetEnvironmentVariableW(_In_ LPCWSTR<br>_In_opt_ LPCWSTR lpValue)                                   |
| uiui  | UINT | SetErrorMode(_In_ UINT uMode)                                                                       |
| ti    | BOOL | SetEvent(_In_ HANDLE hEvent)                                                                        |
| tii   | VOID | SetEventWhenCallbackReturns(_Inout_<br>PTP_CALLBACK_INSTANCE pci, _In_ H                            |
| i     | VOID | SetFileApisToANSI(void)                                                                             |
| i     | VOID | SetFileApisToANSI(void)                                                                             |
| i     | VOID | SetFileApisToOEM(void)                                                                              |
| i     | VOID | SetFileApisToOEM(void)                                                                              |
| suii  | BOOL | SetFileAttributes(_In_ LPCTSTR lpFileNam<br>DWORD dwFileAttributes)                                 |
| aii   | BOOL | SetFileAttributesA(_In_ LPCSTR lpFileNam<br>DWORD dwFileAttributes)                                 |
| suiti | BOOL | SetFileAttributesTransacted(_In_ LPCTSTR<br>_In_ DWORD dwFileAttributes, _In_ HANI<br>hTransaction) |
| auti  | BOOL | SetFileAttributesTransactedA(_In_ LPCSTR<br>_In_ DWORD dwFileAttributes, _In_ HANI<br>hTransaction) |
| wuiti | BOOL | SetFileAttributesTransactedW(_In_ LPCWST<br>_In_ DWORD dwFileAttributes, _In_ HANI<br>hTransaction) |
| wuii  | BOOL | SetFileAttributesW(_In_ LPCWSTR lpFileN<br>DWORD dwFileAttributes)                                  |
|       |      | SetFileBandwidthReservation(_In_ HANDL                                                              |

|          |       |                                                                                                                                                                 |
|----------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiiitti | BOOL  | DWORD nPeriodMilliseconds, _In_ DWORD nBytesPerPeriod, _In_ BOOL bDiscardable, LPDWORD lpTransferSize, _Out_ LPDWORD lpNumOutstandingRequests)                  |
| tuci     | BOOL  | SetFileCompletionNotificationModes(_In_ HANDLE FileHandle, _In_ UCHAR Flags)                                                                                    |
| tituii   | BOOL  | SetFileInformationByHandle(_In_ HANDLE FileHandle, _In_ FILE_INFO_BY_HANDLE_CLASS FileInformationClass, _In_ LPVOID lpFileInformation, _In_ DWORD dwBufferSize) |
| ttuii    | BOOL  | SetFileIoOverlappedRange(_In_ HANDLE FileHandle, _In_ PULONG OverlappedRangeStart, _In_ ULONG OverlappedRangeEnd)                                               |
| tuituiui | DWORD | SetFilePointer(_In_ HANDLE hFile, _In_ LONG DistanceToMove, _Inout_opt_ PLONG lpDistanceToMoveHigh, _In_ DWORD dwMoveMethod)                                    |
| ti6tiii  | BOOL  | SetFilePointerEx(_In_ HANDLE hFile, _In_ LARGE_INTEGER liDistanceToMove, _Out_opt_ PLARGE_INTEGER lpNewFilePointer, _In_ DWORD dwMoveMethod)                    |
| tsi      | BOOL  | SetFileShortName(_In_ HANDLE hFile, _In_ LPCTSTR lpShortName)                                                                                                   |
| tai      | BOOL  | SetFileShortNameA(_In_ HANDLE hFile, _In_ LPCTSTR lpShortName)                                                                                                  |
| twi      | BOOL  | SetFileShortNameW(_In_ HANDLE hFile, _In_ LPCTSTR lpShortName)                                                                                                  |
| ttti     | BOOL  | SetFileTime(_In_ HANDLE hFile, _In_opt_ FILETIME *lpCreationTime, _In_opt_ const FILETIME *lpLastAccessTime, _In_opt_ const FILETIME *lpLastWriteTime)          |
| ti6i     | BOOL  | SetFileValidData(_In_ HANDLE hFile, _In_ ULONGLONG ValidDataLength)                                                                                             |
| sstuii   | BOOL  | SetFirmwareEnvironmentVariable(_In_ LPCWSTR lpName, _In_ LPCTSTR lpGuid, _In_ PVOID pBuffer, DWORD nSize)                                                       |
| aatuii   | BOOL  | SetFirmwareEnvironmentVariableA(_In_ LPCWSTR lpName, _In_ LPCTSTR lpGuid, _In_ PVOID pBuffer, DWORD nSize)                                                      |
| wwtiii   | BOOL  | SetFirmwareEnvironmentVariableW(_In_ LPCWSTR lpName, _In_ LPCWSTR lpGuid, _In_ PVOID pBuffer, DWORD nSize)                                                      |
| uiui     | UINT  | SetHandleCount(UINT uNumber)                                                                                                                                    |
| tuiiii   | BOOL  | SetHandleInformation(_In_ HANDLE hObject, _In_ DWORD dwMask, _In_ DWORD dwFlags)                                                                                |
|          |       | SetInformationJobObject(_In_ HANDLE hJobObject, _In_ JOBOBJECT_INFORMATION_CLASS JobObjectInformationClass, _In_ LPVOID lpInformation, _In_ DWORD dwLength)     |

|         |      |                                                                                                                                                        |
|---------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuituii | BOOL | JOBOBJECTINFOCLASS JobObjectInfoClass, LPVOID lpJobObjectInfo, _In_ DWORD cbJobObjectInfoLength)                                                       |
| uii     | VOID | SetLastError(_In_ DWORD dwErrCode)                                                                                                                     |
| uiuisi  | BOOL | SetLocaleInfo(_In_ LCID Locale, _In_ LCTYPE Category, _In_ LPCTSTR lpLCData)                                                                           |
| uiuiiai | BOOL | SetLocaleInfoA(_In_ LCID Locale, _In_ LCTYPE Category, _In_ LPCSTR lpLCData)                                                                           |
| uiuiwi  | BOOL | SetLocaleInfoW(_In_ LCID Locale, _In_ LCTYPE Category, _In_ LPCWSTR lpLCData)                                                                          |
| ti      | BOOL | SetLocalTime(_In_ const SYSTEMTIME *lpLocalTime)                                                                                                       |
| tuii    | BOOL | SetMailslotInfo(_In_ HANDLE hMailslot, _In_ DWORD dwMailslotFlags, _In_ DWORD lReadTimeout)                                                            |
| tuii    | BOOL | SetMessageWaitingIndicator(HANDLE hMsgIndicator, ULONG ulMsgCount)                                                                                     |
| tttti   | BOOL | SetNamedPipeHandleState(_In_ HANDLE hNamedPipe, _In_opt_ LPDWORD lpMode, _In_opt_ LPDWORD lpMaxCollectionCount, _In_opt_ LPDWORD lpCollectDataTimeout) |
| tuii    | BOOL | SetPriorityClass(_In_ HANDLE hProcess, _In_ DWORD dwPriorityClass)                                                                                     |
| tti     | BOOL | SetProcessAffinityMask(_In_ HANDLE hProcess, _In_ DWORD_PTR dwProcessAffinityMask)                                                                     |
| tuii    | BOOL | SetProcessAffinityUpdateMode(_In_ HANDLE hProcess, _In_ DWORD dwFlags)                                                                                 |
| uii     | BOOL | SetProcessDEPPolicy(_In_ DWORD dwFlags)                                                                                                                |
| uiwti   | BOOL | SetProcessPreferredUILanguages(_In_ DWORD dwFlags, _In_opt_ PCZZWSTR pwszLanguagesBuffer, _In_ PULONG pulNumLanguages)                                 |
| tii     | BOOL | SetProcessPriorityBoost(_In_ HANDLE hProcess, _In_ BOOL DisablePriorityBoost)                                                                          |
| uiuii   | BOOL | SetProcessShutdownParameter(_In_ DWORD dwFlags, _In_ DWORD dwFlags)                                                                                    |
| ttti    | BOOL | SetProcessWorkingSetSize(_In_ HANDLE hProcess, _In_ SIZE_T dwMinimumWorkingSetSize, _In_ SIZE_T dwMaximumWorkingSetSize)                               |
| tttuii  | BOOL | SetProcessWorkingSetSizeEx(_In_ HANDLE hProcess, _In_ SIZE_T dwMinimumWorkingSetSize, _In_ SIZE_T dwMaximumWorkingSetSize, _In_ DWORD dwFlags)         |
| uii     | BOOL | SetSearchPathMode(_In_ DWORD Flags)                                                                                                                    |
| uiti    | BOOL | SetStdHandle(_In_ DWORD nStdHandle, _In_ HANDLE hHandle)                                                                                               |

|             |                 |                                                                                                                                                            |
|-------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuii       | BOOL            | SetSystemFileCacheSize(_In_ SIZE_T MinimumFileCacheSize, _In_ SIZE_T MaximumFileCacheSize, _In_ DWORD Flag)                                                |
| iii         | BOOL            | SetSystemPowerState(_In_ BOOL fSuspend, fForce)                                                                                                            |
| ti          | BOOL            | SetSystemTime(_In_ const SYSTEMTIME *)                                                                                                                     |
| uiii        | BOOL            | SetSystemTimeAdjustment(_In_ DWORD dwTimeAdjustment, _In_ BOOL bTimeAdjustment)                                                                            |
| tuitui      | DWORD           | SetTapeParameters(_In_ HANDLE hDevice, dwOperation, _In_ LPVOID lpTapeInformation)                                                                         |
| tuiuiuiuiui | DWORD           | SetTapePosition(_In_ HANDLE hDevice, _In_ dwPositionMethod, _In_ DWORD dwPartition, _In_ DWORD dwOffsetLow, _In_ DWORD dwOffsetHigh, _In_ BOOL bImmediate) |
| ttt         | DWORD_PTR       | SetThreadAffinityMask(_In_ HANDLE hThread, DWORD_PTR dwThreadAffinityMask)                                                                                 |
| tti         | BOOL            | SetThreadContext(_In_ HANDLE hThread, _In_ CONTEXT *lpContext)                                                                                             |
| uiti        | BOOL            | SetThreadErrorMode(_In_ DWORD dwNewMode, LPDWORD lpOldMode)                                                                                                |
| uiui        | EXECUTION_STATE | SetThreadExecutionState(_In_ EXECUTION_STATE esFlags)                                                                                                      |
| ttti        | BOOL            | SetThreadGroupAffinity(_In_ HANDLE hThread, _In_ GROUP_AFFINITY *GroupAffinity, _Out_ GROUP_AFFINITY *PreviousGroupAffinity)                               |
| tuiui       | DWORD           | SetThreadIdealProcessor(_In_ HANDLE hThread, DWORD dwIdealProcessor)                                                                                       |
| ttti        | BOOL            | SetThreadIdealProcessorEx(_In_ HANDLE hThread, _In_ PROCESSOR_NUMBER lpIdealProcessor, _In_ PROCESSOR_NUMBER lpPreviousIdealProcessor)                     |
| uii         | BOOL            | SetThreadLocale(_In_ LCID Locale)                                                                                                                          |
| tti         | BOOL            | SetThreadpoolStackInformation(_Inout_ PTP_POOL_STACK_INFORMATION)                                                                                          |
| tuii        | VOID            | SetThreadpoolThreadMaximum(_Inout_ PTP_POOL _In_ DWORD cthrdMost)                                                                                          |
| tuii        | BOOL            | SetThreadpoolThreadMinimum(_Inout_ PTP_POOL _In_ DWORD cthrdMic)                                                                                           |
| ttuiuii     | VOID            | SetThreadpoolTimer(_Inout_ PTP_TIMER pTimer, PFILETIME pftDueTime, _In_ DWORD msDueTime, _In_ DWORD msWindowLength)                                        |
| ttti        | VOID            | SetThreadpoolWait(_Inout_ PTP_WAIT pWaitObject, HANDLE h, _In_opt_ PFILETIME pftTimeOut)                                                                   |

|            |                              |                                                                                                                                                                                                               |
|------------|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiwti      | BOOL                         | SetThreadPreferredUILanguages(_In_ DWORD _In_opt_ PCZZWSTR pwszLanguagesBuffer, PULONG pulNumLanguages)                                                                                                       |
| tii        | BOOL                         | SetThreadPriority(_In_ HANDLE hThread, _In_ int nPriority)                                                                                                                                                    |
| tii        | BOOL                         | SetThreadPriorityBoost(_In_ HANDLE hThread, _In_ bool DisablePriorityBoost)                                                                                                                                   |
| ti         | BOOL                         | SetThreadStackGuarantee(_Inout_ PULONG StackSizeInBytes)                                                                                                                                                      |
| uhuh       | LANGID                       | SetThreadUILanguage(_In_ LANGID LangID)                                                                                                                                                                       |
| tttuiuitt  | HANDLE                       | SetTimerQueueTimer(HANDLE TimerQueue, WAITORTIMERCALLBACK Callback, _In_ Parameter, DWORD DueTime, DWORD Period, _In_ bool PreferIo)                                                                          |
| ti         | BOOL                         | SetTimeZoneInformation(_In_ const TIME_ZONE_INFORMATION *lpTimeZoneInformation)                                                                                                                               |
| tt         | LPTOP_LEVEL_EXCEPTION_FILTER | SetUnhandledExceptionFilter(_In_ LPTOP_LEVEL_EXCEPTION_FILTER lpTopLevelExceptionFilter)                                                                                                                      |
| tuiuii     | BOOL                         | SetupComm(_In_ HANDLE hFile, _In_ DWORD dwInQueue, _In_ DWORD dwOutQueue)                                                                                                                                     |
| uii        | BOOL                         | SetUserGeoID(_In_ GEOID GeoId)                                                                                                                                                                                |
| ssi        | BOOL                         | SetVolumeLabel(_In_opt_ LPCTSTR lpRootPath, _In_opt_ LPCTSTR lpVolumeName)                                                                                                                                    |
| aa         | BOOL                         | SetVolumeLabelA(_In_opt_ LPCSTR lpRootPath, _In_opt_ LPCSTR lpVolumeName)                                                                                                                                     |
| wwi        | BOOL                         | SetVolumeLabelW(_In_opt_ LPCWSTR lpRootPath, _In_opt_ LPCWSTR lpVolumeName)                                                                                                                                   |
| ssi        | BOOL                         | SetVolumeMountPoint(_In_ LPCTSTR lpVolumeMountPoint, _In_ LPCTSTR lpVolumeName)                                                                                                                               |
| aa         | BOOL                         | SetVolumeMountPointA(_In_ LPCSTR lpVolumeMountPoint, _In_ LPCSTR lpVolumeName)                                                                                                                                |
| wwi        | BOOL                         | SetVolumeMountPointW(_In_ LPCWSTR lpVolumeMountPoint, _In_ LPCWSTR lpVolumeName)                                                                                                                              |
| ttuittii   | BOOL                         | SetWaitableTimer(_In_ HANDLE hTimer, _In_ LARGE_INTEGER *pDueTime, _In_ LONG lTolerableDelay, _In_opt_ PTIMERAPCROUTINE pfnCompletionRoutine, _In_opt_ LPVOID lpArgToCompletionRoutine, _In_ bool fResume)    |
| ttuitttiii | BOOL                         | SetWaitableTimerEx(_In_ HANDLE hTimer, _In_ LARGE_INTEGER *lpDueTime, _In_ LONG lTolerableDelay, _In_opt_ PTIMERAPCROUTINE pfnCompletionRoutine, _In_opt_ LPVOID lpArgToCompletionRoutine, _In_ bool fResume) |

|         |       |                                                                                                                                                                      |
|---------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |       | PREASON_CONTEXT WakeContext, _In_ TolerableDelay)                                                                                                                    |
| tui6i   | BOOL  | SetXStateFeaturesMask(_Inout_ PCONTEXT DWORD64 FeatureMask)                                                                                                          |
| ttuiui  | DWORD | SignalObjectAndWait(_In_ HANDLE hObject HANDLE hObjectToWaitOn, _In_ DWORD _In_ BOOL bAlertable)                                                                     |
| ttui    | DWORD | SizeofResource(_In_opt_ HMODULE hModule HRSRC hResInfo)                                                                                                              |
| uii     | VOID  | Sleep(_In_ DWORD dwMilliseconds)                                                                                                                                     |
| ttuii   | BOOL  | SleepConditionVariableCS(_Inout_ PCONDITION_VARIABLE ConditionVariable PCRITICAL_SECTION CriticalSection, _In_ dwMilliseconds)                                       |
| ttuiiii | BOOL  | SleepConditionVariableSRW(_Inout_ PCONDITION_VARIABLE ConditionVariable PSRWLOCK SRWLock, _In_ DWORD dwM _In_ ULONG Flags)                                           |
| uiiii   | DWORD | SleepEx(_In_ DWORD dwMilliseconds, _In_ bAlertable)                                                                                                                  |
| ti      | VOID  | StartThreadpools(_Inout_ PTP_IO pio)                                                                                                                                 |
| ti      | VOID  | SubmitThreadpoolWork(_Inout_ PTP_WORK)                                                                                                                               |
| tui     | DWORD | SuspendThread(_In_ HANDLE hThread)                                                                                                                                   |
| ti      | VOID  | SwitchToFiber(_In_ LPVOID lpFiber)                                                                                                                                   |
| i       | BOOL  | SwitchToThread(void)                                                                                                                                                 |
| tii     | BOOL  | SystemTimeToFileTime(_In_ const SYSTEMTIME *lpSystemTime, _Out_ LPFILETIME lpFileTime)                                                                               |
| ttti    | BOOL  | SystemTimeToTzSpecificLocalTime(_In_opt_ LPTIME_ZONE_INFORMATION lpTimeZoneInformation LPSYSTEMTIME lpUniversalTime, _Out_ LPSYSTEMTIME lpLocalTime)                 |
| ttti    | BOOL  | SystemTimeToTzSpecificLocalTimeEx(_In_ DYNAMIC_TIME_ZONE_INFORMATION *lpTimeZoneInformation, _In_ const SYSTEMTIME *lpUniversalTime, _Out_ LPSYSTEMTIME lpLocalTime) |
| tuii    | BOOL  | TerminateJobObject(_In_ HANDLE hJob, _In_ uExitCode)                                                                                                                 |
| tuii    | BOOL  | TerminateProcess(_In_ HANDLE hProcess, _In_ uExitCode)                                                                                                               |
| tuii    | BOOL  | TerminateThread(_Inout_ HANDLE hThread, _In_ dwExitCode)                                                                                                             |
| tii     | BOOL  | Thread32First(_In_ HANDLE hSnapshot, _In_ LPDWORD lpThreadId)                                                                                                        |

|     |         |                                                                                                                                                                                                   |
|-----|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     |         | LPTHREADENTRY32 lpte)                                                                                                                                                                             |
| ttd | BOOL    | Thread32Next(_In_ HANDLE hSnapshot, _Out_ LPTHREADENTRY32 lpte)                                                                                                                                   |
| td  | DWORD   | TlsAlloc(void)                                                                                                                                                                                    |
| td  | BOOL    | TlsFree(_In_ DWORD dwTlsIndex)                                                                                                                                                                    |
| td  | LPVOID  | TlsGetValue(_In_ DWORD dwTlsIndex)                                                                                                                                                                |
| td  | BOOL    | TlsSetValue(_In_ DWORD dwTlsIndex, _In_ lpTlsValue)                                                                                                                                               |
| td  | BOOL    | Toolhelp32ReadProcessMemory(_In_ DWORD th32ProcessID, _In_ LPCVOID lpBaseAddress, LPVOID lpBuffer, _In_ SIZE_T cbRead, _Out_ lpNumberOfBytesRead)                                                 |
| td  | BOOL    | TransactNamedPipe(_In_ HANDLE hNamedPipe, LPVOID lpInBuffer, _In_ DWORD nInBufferSize, LPVOID lpOutBuffer, _In_ DWORD nOutBufferSize, LPDWORD lpBytesRead, _Inout_opt_ LPOVERLAPPED lpOverlapped) |
| td  | BOOL    | TransmitCommChar(_In_ HANDLE hFile, _In_ char c)                                                                                                                                                  |
| td  | BOOLEAN | TryAcquireSRWLockExclusive(_Inout_ PSRWLOCK SRWLock)                                                                                                                                              |
| td  | BOOLEAN | TryAcquireSRWLockShared(_Inout_ PSRWLOCK SRWLock)                                                                                                                                                 |
| td  | BOOL    | TryEnterCriticalSection(_Inout_ LPCRITICAL_SECTION lpCriticalSection)                                                                                                                             |
| td  | BOOL    | TrySubmitThreadpoolCallback(_In_ PTP_SIMPLE_CALLBACK pfns, _Inout_opt_ PVOID pvContext, _In_opt_ PTP_CALLBACK_ENVIRON pcb)                                                                        |
| td  | BOOL    | TzSpecificLocalTimeToSystemTime(_In_opt_ LPTIME_ZONE_INFORMATION lpTimeZoneInformation, _In_ LPSYSTEMTIME lpLocalTime, _Out_ LPSYSTEMTIME lpUniversalTime)                                        |
| td  | BOOL    | TzSpecificLocalTimeToSystemTimeEx(_In_ DYNAMIC_TIME_ZONE_INFORMATION *lpTimeZoneInformation, _In_ const SYSTEMTIME *lpLocalTime, _Out_ LPSYSTEMTIME lpUniversalTime)                              |
| td  | LONG    | UnhandleExceptionFilter(_In_ struct _EXCEPTION_POINTERS *ExceptionInfo)                                                                                                                           |
| td  | BOOL    | UnlockFile(_In_ HANDLE hFile, _In_ DWORD dwFileOffsetLow, _In_ DWORD dwFileOffsetHigh, _In_ DWORD nNumberOfBytesToUnlockLow, _In_ DWORD nNumberOfBytesToUnlockHigh)                               |
| td  | BOOL    | UnlockFileEx(_In_ HANDLE hFile, _Reserved_ dwReserved, _In_ DWORD nNumberOfBytesToUnlock)                                                                                                         |

|           |         |                                                                                                                                                                                                    |
|-----------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |         | _In_ DWORD nNumberOfBytesToUnlockH<br>LPOVERLAPPED lpOverlapped)                                                                                                                                   |
| ti        | BOOL    | UnmapViewOfFile(_In_ LPCVOID lpBaseA                                                                                                                                                               |
| i         | HRESULT | UnregisterApplicationRecoveryCallback(voi                                                                                                                                                          |
| i         | HRESULT | UnregisterApplicationRestart(void)                                                                                                                                                                 |
| ti        | BOOL    | UnregisterWait(_In_ HANDLE WaitHandle)                                                                                                                                                             |
| tii       | BOOL    | UnregistersWaitEx(_In_ HANDLE WaitHand<br>HANDLE CompletionEvent)                                                                                                                                  |
| ti        | BOOL    | UpdateCalendarDayOfWeek(_Inout_ LPCAI<br>lpCalDateTime)                                                                                                                                            |
| tuitttti  | BOOL    | UpdateProcThreadAttribu(_Inout_<br>LPPROC_THREAD_ATTRIBUTE_LIST lp<br>_In_ DWORD dwFlags, _In_ DWORD_PTR<br>PVOID lpValue, _In_ SIZE_T cbSize, _Out_<br>lpPreviousValue, _In_opt_ PSIZE_T lpReturn |
| tssuhtuii | BOOL    | UpdateResource(_In_ HANDLE hUpdate, _I<br>lpType, _In_ LPCTSTR lpName, _In_ WOR<br>_In_opt_ LPVOID lpData, _In_ DWORD cb                                                                           |
| taauhtuii | BOOL    | UpdateResourceA(_In_ HANDLE hUpdate,<br>lpType, _In_ LPCSTR lpName, _In_ WORD<br>_In_opt_ LPVOID lpData, _In_ DWORD cb                                                                             |
| twwuhtuii | BOOL    | UpdateResourceW(_In_ HANDLE hUpdate,<br>lpType, _In_ LPCWSTR lpName, _In_ WOF<br>_In_opt_ LPVOID lpData, _In_ DWORD cb                                                                             |
| uiwiwii   | BOOL    | VerifyScripts(_In_ DWORD dwFlags, _In_ I<br>lpLocaleScripts, _In_ int cchLocaleScripts, _<br>lpTestScripts, _In_ int cchTestScripts)                                                               |
| tuiui6i   | BOOL    | VerifyVersionInfo(_In_ LPOSVERSIONINF<br>lpVersionInfo, _In_ DWORD dwTypeMask,<br>DWORDLONG dwlConditionMask)                                                                                      |
| tuiui6i   | BOOL    | VerifyVersionInfoA(_In_ LPOSVERSIONIN<br>lpVersionInfo, _In_ DWORD dwTypeMask,<br>DWORDLONG dwlConditionMask)                                                                                      |
| tuiui6i   | BOOL    | VerifyVersionInfoW(_In_ LPOSVERSIONIN<br>lpVersionInfo, _In_ DWORD dwTypeMask,<br>DWORDLONG dwlConditionMask)                                                                                      |
| uisuiui   | DWORD   | VerLanguageName(_In_ DWORD wLang, _<br>szLang, _In_ DWORD cchLang)                                                                                                                                 |
| uiauiui   | DWORD   | VerLanguageNameA(_In_ DWORD wLang,<br>szLang, _In_ DWORD cchLang)                                                                                                                                  |
| uiwuiui   | DWORD   | VerLanguageNameW(_In_ DWORD wLang,<br>LPWSTR szLang, _In_ DWORD cchLang)                                                                                                                           |
|           |         | VerSetConditionMask(_In_ ULONGLONG                                                                                                                                                                 |

|             |           |                                                                                                                                                |
|-------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| ui6uiucui6  | ULONGLONG | dwConditionMask, _In_ DWORD dwTypeB<br>BYTE dwConditionMask)                                                                                   |
| ttuiuit     | LPVOID    | VirtualAlloc(_In_opt_ LPVOID lpAddress, _<br>dwSize, _In_ DWORD flAllocationType, _In_<br>flProtect)                                           |
| tttuiuuit   | LPVOID    | VirtualAllocEx(_In_ HANDLE hProcess, _In_<br>lpAddress, _In_ SIZE_T dwSize, _In_ DWO<br>flAllocationType, _In_ DWORD flProtect)                |
| tttuiuuiuit | LPVOID    | VirtualAllocExNuma(_In_ HANDLE hProce<br>LPVOID lpAddress, _In_ SIZE_T dwSize, _<br>flAllocationType, _In_ DWORD flProtect, _<br>nndPreferred) |
| ttuii       | BOOL      | VirtualFree(_In_ LPVOID lpAddress, _In_ S<br>_In_ DWORD dwFreeType)                                                                            |
| tttuii      | BOOL      | VirtualFreeEx(_In_ HANDLE hProcess, _In_<br>lpAddress, _In_ SIZE_T dwSize, _In_ DWO<br>dwFreeType)                                             |
| titi        | BOOL      | VirtualLock(_In_ LPVOID lpAddress, _In_ S                                                                                                      |
| ttuiti      | BOOL      | VirtualProtect(_In_ LPVOID lpAddress, _In_<br>dwSize, _In_ DWORD flNewProtect, _Out_<br>lpflOldProtect)                                        |
| tttuiti     | BOOL      | VirtualProtectEx(_In_ HANDLE hProcess, _<br>lpAddress, _In_ SIZE_T dwSize, _In_ DWO<br>flNewProtect, _Out_ PDWORD lpflOldProte                 |
| tttt        | SIZE_T    | VirtualQuery(_In_opt_ LPCVOID lpAddress<br>PMEMORY_BASIC_INFORMATION lpBu<br>SIZE_T dwLength)                                                  |
| ttttt       | SIZE_T    | VirtualQueryEx(_In_ HANDLE hProcess, _I<br>LPCVOID lpAddress, _Out_<br>PMEMORY_BASIC_INFORMATION lpBu<br>SIZE_T dwLength)                      |
| titi        | BOOL      | VirtualUnlock(_In_ LPVOID lpAddress, _In_<br>dwSize)                                                                                           |
| ttti        | BOOL      | WaitCommEvent(_In_ HANDLE hFile, _Out_<br>lpEvtMask, _In_ LPOVERLAPPED lpOverl                                                                 |
| tuii        | BOOL      | WaitForDebugEvent(_Out_ LPDEBUG_EVI<br>lpDebugEvent, _In_ DWORD dwMilliseconds)                                                                |
| uitiuui     | DWORD     | WaitForMultipleObjects(_In_ DWORD nCot<br>HANDLE *lpHandles, _In_ BOOL bWaitAll<br>dwMilliseconds)                                             |
| uitiuuii    | DWORD     | WaitForMultipleObjectsEx(_In_ DWORD nC<br>HANDLE *lpHandles, _In_ BOOL bWaitAll<br>dwMilliseconds, _In_ BOOL bAlertable)                       |
| tuiui       | DWORD     | WaitForSingleObject(_In_ HANDLE hHandl                                                                                                         |

|             |         |                                                                                                                                                                                                       |
|-------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |         | DWORD dwMilliseconds)                                                                                                                                                                                 |
| tuiui       | DWORD   | WaitForSingleObjectEx(_In_ HANDLE hHa<br>DWORD dwMilliseconds, _In_ BOOL bAle                                                                                                                         |
| tii         | VOID    | WaitForThreadpoolToCallbacks(_Inout_ PTP<br>BOOL fCancelPendingCallbacks)                                                                                                                             |
| tii         | VOID    | WaitForThreadpoolTimerCallbacks(_Inout_ P<br>_In_ BOOL fCancelPendingCallbacks)                                                                                                                       |
| tii         | VOID    | WaitForThreadpoolWaitCallbacks(_Inout_ P<br>_In_ BOOL fCancelPendingCallbacks)                                                                                                                        |
| tii         | VOID    | WaitForThreadpoolWorkCallbacks(_Inout_ P<br>pwk, _In_ BOOL fCancelPendingCallbacks)                                                                                                                   |
| suii        | BOOL    | WaitNamedPipe(_In_ LPCTSTR lpNamedPi<br>DWORD nTimeOut)                                                                                                                                               |
| auui        | BOOL    | WaitNamedPipeA(_In_ LPCSTR lpNamedPi<br>DWORD nTimeOut)                                                                                                                                               |
| wuii        | BOOL    | WaitNamedPipeW(_In_ LPCWSTR lpName<br>DWORD nTimeOut)                                                                                                                                                 |
| ti          | VOID    | WakeAllConditionVariable(_Inout_ P<br>PCONDITION_VARIABLE ConditionVari                                                                                                                               |
| ti          | VOID    | WakeConditionVariable(_Inout_ PCONDI<br>ConditionVariable)                                                                                                                                            |
| tii         | HRESULT | WerGetFlags(_In_ HANDLE hProcess, _Out<br>pdwFlags)                                                                                                                                                   |
| wuiiii      | HRESULT | WerRegisterFile(_In_ PCWSTR pwzFile, _In<br>WER_REGISTER_FILE_TYPE regFileType<br>dwFlags)                                                                                                            |
| tuii        | HRESULT | WerRegisterMemoryBlock(_In_ PVOID pvA<br>DWORD dwSize)                                                                                                                                                |
| wti         | HRESULT | WerRegisterRuntimeExceptionModule(_In_<br>pwszOutOfProcessCallbackDll, _In_opt_ PV                                                                                                                    |
| uii         | HRESULT | WerSetFlags(_In_ DWORD dwFlags)                                                                                                                                                                       |
| wi          | HRESULT | WerUnregisterFile(_In_ PCWSTR pwzFilePa                                                                                                                                                               |
| ti          | HRESULT | WerUnregisterMemoryBlock(_In_ PVOID p                                                                                                                                                                 |
| wti         | HRESULT | WerUnregisterRuntimeExceptionModule(_In<br>pwszOutOfProcessCallbackDll, _In_opt_ PV                                                                                                                   |
| uiuiwiaiati | int     | WideCharToMultiByte(_In_ UINT CodePage<br>dwFlags, _In_ LPCWSTR lpWideCharStr, _I<br>cchWideChar, _Out_opt_ LPSTR lpMultiByt<br>cbMultiByte, _In_opt_ LPCSTR lpDefaultCh<br>LPBOOL lpUsedDefaultChar) |
| aiui        | UINT    | WinExec(_In_ LPCSTR lpCmdLine, _In_ UI                                                                                                                                                                |
|             |         |                                                                                                                                                                                                       |

|          |         |                                                                                                                                                                          |
|----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti       | BOOL    | Wow64DisableWow64FsRedirection(_Out_ *OldValue)                                                                                                                          |
| ucuc     | BOOLEAN | Wow64EnableWow64FsRedirection(_In_ BOOL Wow64FsEnableRedirection)                                                                                                        |
| titi     | BOOL    | Wow64GetThreadContext(_In_ HANDLE hThread, _Out_ PWOW64_CONTEXT lpContext)                                                                                               |
| tuiti    | BOOL    | Wow64GetThreadSelectorEntry(_In_ HANDLE hThread, _In_ DWORD dwSelector, _Out_ PWOW64_THREAD_SELECTOR_ENTRY lpSelectorEntry)                                              |
| ti       | BOOL    | Wow64RevertWow64FsRedirection(_In_ PVOID OldValue)                                                                                                                       |
| titi     | BOOL    | Wow64SetThreadContext(_In_ HANDLE hThread, const WOW64_CONTEXT *lpContext)                                                                                               |
| tui      | DWORD   | Wow64SuspendThread(_In_ HANDLE hThread)                                                                                                                                  |
| ttuitti  | BOOL    | WriteConsole(_In_ HANDLE hConsoleOutput, _In_ VOID *lpBuffer, _In_ DWORD nNumberOfBytesToWrite, _Out_ LPDWORD lpNumberOfCharsWritten, LPVOID lpReserved)                 |
| ttuitti  | BOOL    | WriteConsoleA(_In_ HANDLE hConsoleOutput, _In_ VOID *lpBuffer, _In_ DWORD nNumberOfBytesToWrite, _Out_ LPDWORD lpNumberOfCharsWritten, LPVOID lpReserved)                |
| ttuiti   | BOOL    | WriteConsoleInput(_In_ HANDLE hConsoleOutput, const INPUT_RECORD *lpBuffer, _In_ DWORD nNumberOfEventsToWrite, _Out_ LPDWORD lpNumberOfEventsWritten)                    |
| ttuiti   | BOOL    | WriteConsoleInputA(_In_ HANDLE hConsoleOutput, const INPUT_RECORD *lpBuffer, _In_ DWORD nNumberOfEventsToWrite, _Out_ LPDWORD lpNumberOfEventsWritten)                   |
| ttuiti   | BOOL    | WriteConsoleInputW(_In_ HANDLE hConsoleOutput, const INPUT_RECORD *lpBuffer, _In_ DWORD nNumberOfEventsToWrite, _Out_ LPDWORD lpNumberOfEventsWritten)                   |
| ttuiuiti | BOOL    | WriteConsoleOutput(_In_ HANDLE hConsoleOutput, const CHAR_INFO *lpBuffer, _In_ COORD dwBufferCoord, _In_ COORD dwWriteCoord, _Inout_ PSMALL_RECT lpWriteRegion)          |
| ttuiuiti | BOOL    | WriteConsoleOutputA(_In_ HANDLE hConsoleOutput, const CHAR_INFO *lpBuffer, _In_ COORD dwBufferCoord, _In_ COORD dwWriteCoord, _Inout_ PSMALL_RECT lpWriteRegion)         |
| ttuiuiti | BOOL    | WriteConsoleOutputAttribute(_In_ HANDLE hConsoleOutput, _In_ const WORD *lpAttribute, _In_ DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfAttrsWritten) |
| tsuiuiti | BOOL    | WriteConsoleOutputCharacter(_In_ HANDLE hConsoleOutput, _In_ LPCTSTR lpCharacter, _In_ DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfCharsWritten)     |

|          |      |                                                                                                                                                                                            |
|----------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |      | lpNumberOfCharsWritten)                                                                                                                                                                    |
| tauiuiti | BOOL | WriteConsoleOutputCharacterA(_In_ HANDLE hConsoleOutput, _In_ LPCSTR lpCharacter, DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfCharsWritten)                            |
| twuiuiti | BOOL | WriteConsoleOutputCharacterW(_In_ HANDLE hConsoleOutput, _In_ LPCWSTR lpCharacter, DWORD nLength, _In_ COORD dwWriteCoord, _Out_ LPDWORD lpNumberOfCharsWritten)                           |
| ttuiuiti | BOOL | WriteConsoleOutputW(_In_ HANDLE hConsoleOutput, const CHAR_INFO *lpBuffer, _In_ COORD dwWriteCoord, _In_ COORD dwBufferCoord, _Inout_ PSMDIWRITE lpWriteRegion)                            |
| ttuitti  | BOOL | WriteConsoleW(_In_ HANDLE hConsoleOutput, VOID *lpBuffer, _In_ DWORD nNumberOfBytesToWrite, _Out_ LPDWORD lpNumberOfCharsWritten, LPVOID lpReserved)                                       |
| ttuitti  | BOOL | WriteFile(_In_ HANDLE hFile, _In_ LPCVOID lpBuffer, _In_ DWORD nNumberOfBytesToWrite, _Out_ LPDWORD lpNumberOfBytesWritten, _Inout_ LPOVERLAPPED lpOverlapped)                             |
| ttuitti  | BOOL | WriteFileEx(_In_ HANDLE hFile, _In_opt_ VOID *lpBuffer, _In_ DWORD nNumberOfBytesToWrite, _Inout_ LPOVERLAPPED lpOverlapped, _In_opt_ LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine) |
| ttuitti  | BOOL | WriteFileGather(_In_ HANDLE hFile, _In_ FILE_SEGMENT_ELEMENT aSegmentArray, _In_ DWORD nNumberOfBytesToWrite, _Reserved_ LPVOID lpReserved, _Inout_ LPOVERLAPPED lpOverlapped)             |
| sssi     | BOOL | WritePrivateProfileSection(_In_ LPCTSTR lpSectionName, _In_ LPCTSTR lpString, _In_ LPCTSTR lpFileName)                                                                                     |
| aaai     | BOOL | WritePrivateProfileSectionA(_In_ LPCSTR lpSectionName, _In_ LPCSTR lpString, _In_ LPCSTR lpFileName)                                                                                       |
| wwwi     | BOOL | WritePrivateProfileSectionW(_In_ LPCWSTR lpSectionName, _In_ LPCWSTR lpString, _In_ LPCWSTR lpFileName)                                                                                    |
| ssssi    | BOOL | WritePrivateProfileString(_In_ LPCTSTR lpSectionName, _In_ LPCTSTR lpKeyName, _In_ LPCTSTR lpString, _In_ LPCTSTR lpFileName)                                                              |
| aaaai    | BOOL | WritePrivateProfileStringA(_In_ LPCSTR lpSectionName, _In_ LPCSTR lpKeyName, _In_ LPCSTR lpString, _In_ LPCSTR lpFileName)                                                                 |
| wwwwi    | BOOL | WritePrivateProfileStringW(_In_ LPCWSTR lpSectionName, _In_ LPCWSTR lpKeyName, _In_ LPCWSTR lpString, _In_ LPCWSTR lpFileName)                                                             |

|          |       |                                                                                                                                                                      |
|----------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sstuisi  | BOOL  | <a href="#">WritePrivateProfileStruct</a> (_In_ LPCTSTR lpStruct, _In_ LPCTSTR lpszKey, _In_ LPVOID lpStruct, _In_ LPCTSTR szFile)                                   |
| aatuiiai | BOOL  | <a href="#">WritePrivateProfileStructA</a> (_In_ LPCSTR lpStruct, _In_ LPCSTR lpszKey, _In_ LPVOID lpStruct, _In_ LPCSTR szFile)                                     |
| wwtuiwi  | BOOL  | <a href="#">WritePrivateProfileStructW</a> (_In_ LPCWSTR lpStruct, _In_ LPCWSTR lpszKey, _In_ LPVOID lpStruct, _In_ LPCWSTR szFile)                                  |
| tttiti   | BOOL  | <a href="#">WriteProcessMemory</a> (_In_ HANDLE hProcess, _In_ LPVOID lpBaseAddress, _In_ LPCVOID lpBuffer, _In_ SIZE_T nSize, _Out_ SIZE_T *lpNumberOfBytesWritten) |
| ssi      | BOOL  | <a href="#">WriteProfileSection</a> (_In_ LPCTSTR lpAppName, _In_ LPCTSTR lpString)                                                                                  |
| aaai     | BOOL  | <a href="#">WriteProfileSectionA</a> (_In_ LPCSTR lpAppName, _In_ LPCSTR lpString)                                                                                   |
| wwi      | BOOL  | <a href="#">WriteProfileSectionW</a> (_In_ LPCWSTR lpAppName, _In_ LPCWSTR lpString)                                                                                 |
| sssi     | BOOL  | <a href="#">WriteProfileString</a> (_In_ LPCTSTR lpAppName, _In_ LPCTSTR lpKeyName, _In_ LPCTSTR lpString)                                                           |
| aaai     | BOOL  | <a href="#">WriteProfileStringA</a> (_In_ LPCSTR lpAppName, _In_ LPCSTR lpKeyName, _In_ LPCSTR lpString)                                                             |
| wwwi     | BOOL  | <a href="#">WriteProfileStringW</a> (_In_ LPCWSTR lpAppName, _In_ LPCWSTR lpKeyName, _In_ LPCWSTR lpString)                                                          |
| tuiuiui  | DWORD | <a href="#">WriteTapemark</a> (_In_ HANDLE hDevice, _In_ DWORD dwTapemarkType, _In_ DWORD dwTapemarkValue, _In_ BOOL bImmediate)                                     |
| ui       | DWORD | <a href="#">WTSGetActiveConsoleSessionId</a> (void)                                                                                                                  |
| ti       | BOOL  | <a href="#">ZombifyActCtx</a> (_In_ HANDLE hActCtx)                                                                                                                  |

## Ole32.dll

|           |         |                                                                                                                                                                         |
|-----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti    | HRESULT | BindMoniker(_In_ LPMONIKER pmk, _In_ DWORD grfOpt, _In_ REFIID iidResult, _Out_ LPVOID *ppvResult)                                                                      |
| sti       | HRESULT | CLSIDFromProgID(_In_ LPCOLESTR lpszProgID, _Out_ LPCLSID lpclsid)                                                                                                       |
| sti       | HRESULT | CLSIDFromProgIDEx(_In_ LPCOLESTR lpszProgID, _Out_ LPCLSID lpclsid)                                                                                                     |
| sti       | HRESULT | CLSIDFromString(_In_ LPCOLESTR lpsz, _Out_ LPCLSID pclsid)                                                                                                              |
| ui        | ULONG   | CoAddRefServerProcess(void)                                                                                                                                             |
| titi      | HRESULT | CoAllowSetForegroundWindow(_In_ IUnknown *pUnk, _In_ LPVOID lpvReserved)                                                                                                |
| uiiii     | HRESULT | CoCancelCall(_In_ DWORD dwThreadId, _In_ ULONG ulTimeout)                                                                                                               |
| titi      | HRESULT | CoCopyProxy(_In_ IUnknown *pProxy, _Out_ IUnknown **ppCopy)                                                                                                             |
| titi      | HRESULT | CoCreateFreeThreadedMarshaler(_In_ LPUNKNOWN punkOuter, _Out_ LPUNKNOWN *ppunkMarshal)                                                                                  |
| ti        | HRESULT | CoCreateGuid(_Out_ GUID *pguid)                                                                                                                                         |
| ttuitti   | HRESULT | CoCreateInstance(_In_ REFCLSID rclsid, _In_ LPUNKNOWN pUnkOuter, _In_ DWORD dwClsContext, _In_ REFIID riid, _Out_ LPVOID *ppv)                                          |
| ttuituiti | HRESULT | CoCreateInstanceEx(_In_ REFCLSID rclsid, _In_ IUnknown *punkOuter, _In_ DWORD dwClsCtx, _In_ COSERVERINFO *pServerInfo, _In_ DWORD dwCount, _Inout_ MULTI_QI *pResults) |
| ti        | HRESULT | CoDisableCallCancellation(_In_opt_ LPVOID pReserved)                                                                                                                    |
| uiii      | HRESULT | CoDisconnectContext(_In_ DWORD dwTimeout)                                                                                                                               |
| tuiii     | HRESULT | CoDisconnectObject(_In_ LPUNKNOWN pUnk, _In_ DWORD dwReserved)                                                                                                          |
| uhuhti    | BOOL    | CoDosDateTimeToFileTime(_In_ WORD nDosDate, _In_ WORD nDosTime, _Out_ FILETIME *lpFileTime)                                                                             |
|           |         | CoEnableCallCancellation(_In_opt_ LPVOID                                                                                                                                |

|               |         |                                                                                                                                                                                                                                  |
|---------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti            | HRESULT | pReserved)                                                                                                                                                                                                                       |
| ti            | HRESULT | CoFileTimeNow(_Out_ FILETIME *lpFileTime)                                                                                                                                                                                        |
| ttti          | BOOL    | CoFileTimeToDosDateTime(_In_ FILETIME *lpFileTime, _Out_ LPWORD lpDosDate, _Out_ LPWORD lpDosTime)                                                                                                                               |
| i             | VOID    | CoFreeAllLibraries(void)                                                                                                                                                                                                         |
| ti            | VOID    | CoFreeLibrary(_In_ HINSTANCE hInst)                                                                                                                                                                                              |
| i             | VOID    | CoFreeUnusedLibraries(void)                                                                                                                                                                                                      |
| uiiii         | VOID    | CoFreeUnusedLibrariesEx(_In_ DWORD dwUnloadDelay, _In_ DWORD dwReserved)                                                                                                                                                         |
| tii           | HRESULT | CoGetApartmentType(_Out_ APTTYPE *pAptType, _Out_ APTTYPEQUALIFIER *pAptQualifier)                                                                                                                                               |
| tii           | HRESULT | CoGetCallContext(_In_ REFIID riid, _Out_ void **ppInterface)                                                                                                                                                                     |
| ti            | HRESULT | CoGetCallerTID(_Out_ LPDWORD lpdwTID)                                                                                                                                                                                            |
| uitti         | HRESULT | CoGetCancelObject(_In_ DWORD dwThreadId, _In_ REFIID iid, _Out_ void **ppUnk)                                                                                                                                                    |
| tuittti       | HRESULT | CoGetClassObject(_In_ REFCLSID rclsid, _In_ DWORD dwClsContext, _In_opt_ COSERVERINFO *pServerInfo, _In_ REFIID riid, _Out_ LPVOID *ppv)                                                                                         |
| ti            | HRESULT | CoGetContextToken(_Out_ ULONG_PTR *pToken)                                                                                                                                                                                       |
| ti            | HRESULT | CoGetCurrentLogicalThreadId(_Out_ GUID *pguid)                                                                                                                                                                                   |
| ui            | DWORD   | CoGetCurrentProcess(void)                                                                                                                                                                                                        |
| itti          | HRESULT | CoGetDefaultContext(_In_ APITYPE aptType, _In_ REFIID riid, _Out_ void **ppv)                                                                                                                                                    |
| tttuiuuiwuiti | HRESULT | CoGetInstanceFromFile(_In_opt_ COSERVERINFO *pServerInfo, _In_opt_ CLSID *pClsid, _In_opt_ IUnknown *punkOuter, _In_ DWORD dwClsCtx, _In_ DWORD grfMode, _In_ OLECHAR *pwszName, _In_ DWORD dwCount, _Inout_ MULTI_QI *pResults) |
| tttuituiti    | HRESULT | CoGetInstanceFromIStorage(_In_opt_ COSERVERINFO *pServerInfo, _In_opt_ CLSID *pClsid, _In_opt_ IUnknown *punkOuter, _In_ DWORD dwClsCtx, _In_ struct IStorage *pstg, _In_ DWORD dwCount, _Inout_ MULTI_QI *pResults)             |

|              |         |                                                                                                                                                                                                                                                                                              |
|--------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttti         | HRESULT | CoGetInterceptor(_In_ REFIID iidIntercepted, _In_ IUnknown *punkOuter, _In_ REFIID iid, _Out_ void **ppv)                                                                                                                                                                                    |
| ttti         | HRESULT | CoGetInterfaceAndReleaseStream(_In_ LPSTREAM pStm, _In_ REFIID iid, _Out_ LPVOID *ppv)                                                                                                                                                                                                       |
| uiti         | HRESULT | CoGetMalloc(_In_ DWORD dwMemContext, _Out_ LPMALLOC *ppMalloc)                                                                                                                                                                                                                               |
| tttuitui     | HRESULT | CoGetMarshalSizeMax(_Out_ ULONG *pulSize, _In_ REFIID riid, _In_ LPUNKNOWN pUnk, _In_ DWORD dwDestContext, _In_opt_ LPVOID pvDestContext, _In_ DWORD mshlflags)                                                                                                                              |
| wttti        | HRESULT | CoGetObject(_In_ LPCWSTR pszName, _In_opt_ BIND_OPTS *pBindOptions, _In_ REFIID riid, _Out_ void **ppv)                                                                                                                                                                                      |
| tti          | HRESULT | CoGetObjectContext(_In_ REFIID riid, _Out_ LPVOID *ppv)                                                                                                                                                                                                                                      |
| tti          | HRESULT | CoGetPSClsid(_In_ REFIID riid, _Out_ CLSID *pClsid)                                                                                                                                                                                                                                          |
| ttuituiti    | HRESULT | CoGetStandardMarshal(_In_ REFIID riid, _In_ LPUNKNOWN pUnk, _In_ DWORD dwDestContext, _In_opt_ LPVOID pvDestContext, _In_ DWORD mshlflags, _Out_ LPMARSHAL *ppMarshal)                                                                                                                       |
| tuiti        | HRESULT | CoGetStdMarshalEx(_In_ LPUNKNOWN pUnkOuter, _In_ DWORD smexflags, _Out_ LPUNKNOWN *ppUnkInner)                                                                                                                                                                                               |
| tti          | HRESULT | CoGetTreatAsClass(_In_ REFCLSID clsidOld, _Out_ LPCLSID pClsidNew)                                                                                                                                                                                                                           |
| i            | HRESULT | CoImpersonateClient(void)                                                                                                                                                                                                                                                                    |
| ti           | HRESULT | CoInitialize(_In_opt_ LPVOID pvReserved)                                                                                                                                                                                                                                                     |
| tuii         | HRESULT | CoInitializeEx(_In_opt_ LPVOID pvReserved, _In_ DWORD dwCoInit)                                                                                                                                                                                                                              |
| tuittuuiuiti | HRESULT | CoInitializeSecurity(_In_opt_ PSECURITY_DESCRIPTOR pSecDesc, _In_ LONG cAuthSvc, _In_opt_ SOLE_AUTHENTICATION_SERVICE *asAuthSvc, _In_opt_ void *pReserved1, _In_ DWORD dwAuthnLevel, _In_ DWORD dwImpLevel, _In_opt_ void *pAuthList, _In_ DWORD dwCapabilities, _In_opt_ void *pReserved3) |
|              |         | CoInstall(_In_ IBindCtx *pbc, _In_ DWORD dwFlags, _In_ uCLSPEC *pClassSpec, _In_                                                                                                                                                                                                             |

|          |           |                                                                                                                                                                                                                                                                                    |
|----------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuittwi  | HRESULT   | QUERYCONTEXT *pQuery, _In_ LPWSTR pszCodeBase)                                                                                                                                                                                                                                     |
| si       | HRESULT   | CoValidateRemoteMachineBindings(_In_ LPOLESTR pszMachineName)                                                                                                                                                                                                                      |
| ti       | BOOL      | CoIsHandlerConnected(_In_ LPUNKNOWN pUnk)                                                                                                                                                                                                                                          |
| ti       | BOOL      | CoIsOleIClass(_In_ REFCLSID reclsid)                                                                                                                                                                                                                                               |
| sit      | HINSTANCE | CoLoadLibrary(_In_ LPOLESTR lpszLibName, _In_ BOOL bAutoFree)                                                                                                                                                                                                                      |
| tiii     | HRESULT   | CoLockObjectExternal(_In_ LPUNKNOWN pUnk, _In_ BOOL fLock, _In_ BOOL fLastUnlockReleases)                                                                                                                                                                                          |
| tii      | HRESULT   | CoMarshalHresult(_In_ LPSTREAM pstm, _In_ HRESULT hresult)                                                                                                                                                                                                                         |
| ttuituii | HRESULT   | CoMarshalInterface(_In_ LPSTREAM pStm, _In_ REFIID riid, _In_ LPUNKNOWN pUnk, _In_ DWORD dwDestContext, _In_opt_ LPVOID pvDestContext, _In_ DWORD mshlflags)                                                                                                                       |
| ttti     | HRESULT   | CoMarshalInterThreadInterfaceInStream(_In_ REFIID riid, _In_ LPUNKNOWN pUnk, _Out_ LPSTREAM *ppStm)                                                                                                                                                                                |
| tii      | HRESULT   | CoQueryAuthenticationServices(_Out_ DWORD *pcAuthSvc, _Out_ SOLE_AUTHENTICATION_SERVICE **asAuthSvc)                                                                                                                                                                               |
| ttwttti  | HRESULT   | CoQueryClientBlanket(_Out_opt_ DWORD *pAuthnSvc, _Out_opt_ DWORD *pAuthzSvc, _Out_opt_ OLECHAR **pServerPrincName, _Out_opt_ DWORD *pAuthnLevel, _Out_opt_ DWORD *pImpLevel, _Out_opt_ RPC_AUTHZ_HANDLE *pPrivs, _Inout_opt_ DWORD *pCapabilities)                                 |
| ttwttti  | HRESULT   | CoQueryProxyBlanket(_In_ IUnknown *pProxy, _Out_opt_ DWORD *pwAuthnSvc, _Out_opt_ DWORD *pAuthzSvc, _Out_opt_ OLECHAR **pServerPrincName, _Out_opt_ DWORD *pAuthnLevel, _Out_opt_ DWORD *pImpLevel, _Out_opt_ RPC_AUTH_IDENTITY_HANDLE *pAuthInfo, _Out_opt_ DWORD *pCapabilities) |
| tii      | HRESULT   | CoRegisterChannelHook(_In_ REFGUID ExtensionUuid, _In_ IChannelHook *pChannelHook)                                                                                                                                                                                                 |

|              |         |                                                                                                                                                                                                                                                    |
|--------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiuiti     | HRESULT | CoRegisterClassObject(_In_ REFCLSID rclsid, _In_ LPUNKNOWN pUnk, _In_ DWORD dwClsContext, _In_ DWORD flags, _Out_ LPDWORD lpdwRegister)                                                                                                            |
| tii          | HRESULT | CoRegisterInitializeSpy(_In_ LPINITIALIZESPY pSpy, _Out_ ULARGE_INTEGER *puliCookie)                                                                                                                                                               |
| ti           | HRESULT | CoRegisterMallocSpy(_In_ LPMALLOCSPY pMallocSpy)                                                                                                                                                                                                   |
| tii          | HRESULT | CoRegisterMessageFilter(_In_opt_ LPMESSAGEFILTER lpMessageFilter, _Out_opt_ LPMESSAGEFILTER *lpMessageFilter)                                                                                                                                      |
| tii          | HRESULT | CoRegisterPSClsid(_In_ REFIID riid, _In_ REFCLSID rclsid)                                                                                                                                                                                          |
| ti           | HRESULT | CoRegisterSurrogate(_In_ LPSURROGATE pSurrogate)                                                                                                                                                                                                   |
| ti           | HRESULT | CoReleaseMarshalData(_In_ LPSTREAM pStm)                                                                                                                                                                                                           |
| ui           | ULONG   | CoReleaseServerProcess(void)                                                                                                                                                                                                                       |
| i            | HRESULT | CoResumeClassObjects(void)                                                                                                                                                                                                                         |
| i            | HRESULT | CoRevertToSelf(void)                                                                                                                                                                                                                               |
| uii          | HRESULT | CoRevokeClassObject(_In_ DWORD dwRegister)                                                                                                                                                                                                         |
| i6i          | HRESULT | CoRevokeInitializeSpy(_In_ ULARGE_INTEGER uliCookie)                                                                                                                                                                                               |
| i            | HRESULT | CoRevokeMallocSpy(void)                                                                                                                                                                                                                            |
| ti           | HRESULT | CoSetCancelObject(_In_opt_ IUnknown *pUnk)                                                                                                                                                                                                         |
| tuiiwuiuitui | HRESULT | CoSetProxyBlanket(_In_ IUnknown *pProxy, _In_ DWORD dwAuthnSvc, _In_ DWORD dwAuthzSvc, _In_opt_ OLECHAR *pServerPrincName, _In_ DWORD dwAuthnLevel, _In_ DWORD dwImpLevel, _In_opt_ RPC_AUTH_IDENTITY_HANDLE pAuthInfo, _In_ DWORD dwCapabilities) |
| i            | HRESULT | CoSuspendClassObjects(void)                                                                                                                                                                                                                        |
| tii          | HRESULT | CoSwitchCallContext(_In_opt_ IUnknown *pNewObject, _Out_ IUnknown **ppOldObject)                                                                                                                                                                   |
| tt           | LPVOID  | CoTaskMemAlloc(_In_ SIZE_T cb)                                                                                                                                                                                                                     |
| ti           | VOID    | CoTaskMemFree(_In_opt_ LPVOID pv)                                                                                                                                                                                                                  |
| ttt          | LPVOID  | CoTaskMemRealloc(_In_opt_ LPVOID pv, _In_ SIZE_T cb)                                                                                                                                                                                               |

|          |         |                                                                                                                                          |
|----------|---------|------------------------------------------------------------------------------------------------------------------------------------------|
| i        | HRESULT | CoTestCancel(void)                                                                                                                       |
| titi     | HRESULT | CoTreatAsClass(_In_ REFCLSID clsidOld, _In_ REFCLSID clsidNew)                                                                           |
| i        | VOID    | CoUninitialize(void)                                                                                                                     |
| titi     | HRESULT | CoUnmarshalHresult(_In_ LPSTREAM pstm, _Out_ HRESULT *phresult)                                                                          |
| titi     | HRESULT | CoUnmarshalInterface(_In_ LPSTREAM pstm, _In_ REFIID riid, _Out_ LPVOID *ppv)                                                            |
| uiuiuiti | HRESULT | CoWaitForMultipleHandles(_In_ DWORD dwFlags, _In_ DWORD dwTimeout, _In_ ULONG cHandles, _In_ LPHANDLE pHandles, _Out_ LPDWORD lpdwindex) |
| ti       | HRESULT | CreateAntiMoniker(_Out_ LPMONIKER *ppmk)                                                                                                 |
| uiti     | HRESULT | CreateBindCtx(_In_ DWORD reserved, _Out_ LPBC *ppbc)                                                                                     |
| titi     | HRESULT | CreateClassMoniker(_In_ REFCLSID rclsid, _Out_ LPMONIKER *ppmk)                                                                          |
| ti       | HRESULT | CreateDataAdviseHolder(_Out_ LPDATAADVISEHOLDER *ppDAHlder)                                                                              |
| ttiti    | HRESULT | CreateDataCache(_In_ LPUNKNOWN pUnkOuter, _In_ REFCLSID rclsid, _In_ REFIID iid, _Out_ LPVOID *ppv)                                      |
| stti     | HRESULT | CreateFileMoniker(_In_ LPCOLESTR lpszPathName, _Out_ LPMONIKER *ppmk)                                                                    |
| titi     | HRESULT | CreateGenericComposite(_In_opt_ LPMONIKER pmkFirst, _In_opt_ LPMONIKER pmkRest, _Out_ LPMONIKER *ppmkComposite)                          |
| titi     | int     | CreateLockBytesOnHGlobal(_In_ HGLOBAL hGlobal, _In_ BOOL fDeleteOnRelease, _Out_ ILockBytes **ppLkbyt)                                   |
| sstti    | HRESULT | CreateItemMoniker(_In_ LPCOLESTR lpszDelim, _In_ LPCOLESTR lpszItem, _Out_ LPMONIKER *ppmk)                                              |
| titi     | HRESULT | CreateObjrefMoniker(_In_opt_ LPUNKNOWN punk, _Out_ LPMONIKER *ppmk)                                                                      |
| ti       | HRESULT | CreateOleAdviseHolder(_Out_ LPOLEADVISEHOLDER *ppOAHlder)                                                                                |
| titi     | HRESULT | CreatePointerMoniker(_In_opt_ LPUNKNOWN punk, _Out_ LPMONIKER *ppmk)                                                                     |
| titi     | int     | CreateStreamOnHGlobal(_In_ HGLOBAL hGlobal, _In_ BOOL fDeleteOnRelease, _Out_                                                            |

|             |         |                                                                                                                           |
|-------------|---------|---------------------------------------------------------------------------------------------------------------------------|
|             |         | LPSTREAM *ppstm)                                                                                                          |
| iti         | BOOL    | DllDebugObjectRPCHook (BOOL fTrace, LPORPC_INIT_ARGS lpOrpcInitArgs)                                                      |
| ttuiti      | HRESULT | DoDragDrop (_In_ LPDATAOBJECT pDataObj, _In_ LPDROPSOURCE pDropSource, _In_ DWORD dwOKEffects, _Out_ LPDWORD pdwEffect)   |
| tsi         | HRESULT | FindIdToPropSigName (_In_ const FMTID *pfmtid, _Out_ LPOLESTR pszName)                                                    |
| uiti        | HRESULT | FreePropVariantArray (_In_ ULONG cVariants, _Inout_ PROPVARIANT *rgvars)                                                  |
| sti         | HRESULT | GetClassFile (_In_ LPCOLESTR szFilename, _Out_ CLSID *pclsid)                                                             |
| ti          | int     | GetConvertSig (_In_ IStorage *pStg)                                                                                       |
| tti         | int     | GetHGlobalFromLockBytes (_In_ ILockBytes *pLkbyt, _Out_ HGLOBAL *phglobal)                                                |
| tti         | int     | GetHGlobalFromStream (_In_ IStream *pstm, _Out_ HGLOBAL *phglobal)                                                        |
| uiti        | HRESULT | GetRunningObjectTable (_In_ DWORD reserved, _Out_ LPRUNNINGOBJECTTABLE *pprot)                                            |
| sti         | HRESULT | IIDFromString (_In_ LPCOLESTR lpsz, _Out_ LPIID lpiid)                                                                    |
| titti       | BOOL    | IsAccelerator (_In_ HACCEL hAccel, _In_ int cAccelEntries, _In_ LPMSG lpMsg, _Out_ WORD *lpwCmd)                          |
| tii         | BOOL    | IsEqualGUID (_In_ REFGUID rguid1, _In_ REFGUID rguid2)                                                                    |
| tssti       | HRESULT | MkParseDisplayName (_In_ LPBC pbc, _In_ LPCOLESTR szUserName, _Out_ ULONG *pchEaten, _Out_ LPMONIKER *ppmk)               |
| titi        | HRESULT | MonikerCommonPrefixWith (_In_ LPMONIKER pmkThis, _In_ LPMONIKER pmkOther, _Out_ LPMONIKER *ppmkCommon)                    |
| ttii        | HRESULT | MonikerRelativePathTo (_In_ LPMONIKER pmkSrc, _In_ LPMONIKER pmkDest, _Out_ LPMONIKER *ppmkRelPath, _In_ BOOL dwReserved) |
| tii         | int     | OleConvertIStorageToOLESTREAM (_In_ IStorage *pStg, _Out_ LPOLESTREAM lpolestream)                                        |
| tuhuiuiitti | int     | OleConvertIStorageToOLESTREAMEx (_In_ IStorage *pStg, _In_ CLIPFORMAT cfFormat, _In_ LONG lWidth, _In_ LONG lHeight, _In_ |

|                |         |                                                                                                                                                                                                                                                                                                                                 |
|----------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                |         | DWORD dwSize, _In_ STGMEDIUM pmedium, _Out_ LPOLESTREAM lpolestm)                                                                                                                                                                                                                                                               |
| ttti           | int     | OleConvertOLESTREAMToIStorage(_In_ LPOLESTREAM lpolestream, _Out_ IStorage *pstg, _In_ const DVTARGETDEVICE *ptd)                                                                                                                                                                                                               |
| tttttti        | int     | OleConvertOLESTREAMToIStorageEx(_In_ LPOLESTREAM lpolestm, _Out_ IStorage *pstg, _Out_ CLIPFORMAT *pcfFormat, _Out_ LONG *plWidth, _Out_ LONG *plHeight, _Out_ DWORD *pdwSize, _Out_ STGMEDIUM pmedium)                                                                                                                         |
| ttuittti       | HRESULT | OleCreate(_In_ REFCLSID rclsid, _In_ REFIID riid, _In_ DWORD renderopt, _In_ LPFORMATETC pFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)                                                                                                                                               |
| tttti          | HRESULT | OleCreateDefaultHandler(_In_ REFCLSID clsid, _In_ LPUNKNOWN pUnkOuter, _In_ REFIID riid, _Out_ LPVOID *lpObj)                                                                                                                                                                                                                   |
| ttuittti       | HRESULT | OleCreateEmbeddingHelper(_In_ REFCLSID clsid, _In_ LPUNKNOWN pUnkOuter, _In_ DWORD flags, _In_ LPCLASSFACTORY pCF, _In_ REFIID riid, _Out_ LPVOID *lpObj)                                                                                                                                                                       |
| ttuiuiutttttti | HRESULT | OleCreateEx(_In_ REFCLSID rclsid, _In_ REFIID riid, _In_ DWORD dwFlags, _In_ DWORD renderopt, _In_ ULONG cFormats, _In_ DWORD *rgAdvf, _In_ LPFORMATETC rgFormatEtc, _In_ IAdviseSink *lpAdviseSink, _Out_ DWORD *rgdwConnection, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)                  |
| ttuittti       | HRESULT | OleCreateFromData(_In_ LPDATAOBJECT pSrcDataObj, _In_ REFIID riid, _In_ DWORD renderopt, _In_ LPFORMATETC pFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)                                                                                                                              |
| ttuiuiutttttti | HRESULT | OleCreateFromDataEx(_In_ LPDATAOBJECT pSrcDataObj, _In_ REFIID riid, _In_ DWORD dwFlags, _In_ DWORD renderopt, _In_ ULONG cFormats, _In_ DWORD *rgAdvf, _In_ LPFORMATETC rgFormatEtc, _In_ IAdviseSink *lpAdviseSink, _Out_ DWORD *rgdwConnection, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj) |
|                |         | OleCreateFromFile(_In_ REFCLSID rclsid, _In_ LPCOLESTR lpszFileName, _In_ REFIID riid,                                                                                                                                                                                                                                          |

|                  |         |                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tstuittiti       | HRESULT | <a href="#">_In_ DWORD renderopt, _In_ LPFORMATETC lpFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a>                                                                                                                                                                                                                    |
| tstuiuiuittttiti | HRESULT | <a href="#">OleCreateFromFileEx(_In_ REFCLSID rclsid, _In_ LPCOLESTR lpszFileName, _In_ REFIID riid, _In_ DWORD dwFlags, _In_ DWORD renderopt, _In_ ULONG cFormats, _In_ DWORD *rgAdvf, _In_ LPFORMATETC rgFormatEtc, _In_ IAdviseSink *lpAdviseSink, _Out_ DWORD *rgdwConnection, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a> |
| ttuittiti        | HRESULT | <a href="#">OleCreateLink(_In_ LPMONIKER pmkLinkSrc, _In_ REFIID riid, _In_ DWORD renderopt, _In_ LPFORMATETC lpFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a>                                                                                                                                                         |
| ttuiuiuittttiti  | HRESULT | <a href="#">OleCreateLinkEx(_In_ LPMONIKER pmkLinkSrc, _In_ REFIID riid, _In_ DWORD dwFlags, _In_ DWORD renderopt, _In_ ULONG cFormats, _In_ DWORD *rgAdvf, _In_ LPFORMATETC rgFormatEtc, _In_ IAdviseSink *lpAdviseSink, _Out_ DWORD *rgdwConnection, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a>                             |
| ttuittiti        | HRESULT | <a href="#">OleCreateLinkFromData(_In_ LPDATAOBJECT pSrcDataObj, _In_ REFIID riid, _In_ DWORD renderopt, _In_ LPFORMATETC pFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a>                                                                                                                                              |
| ttuiuiuittttiti  | HRESULT | <a href="#">OleCreateLinkFromDataEx(_In_ LPDATAOBJECT pSrcDataObj, _In_ REFIID riid, _In_ DWORD dwFlags, _In_ DWORD renderopt, _In_ ULONG cFormats, _In_ DWORD *rgAdvf, _In_ LPFORMATETC rgFormatEtc, _In_ IAdviseSink *lpAdviseSink, _Inout_ DWORD *rgdwConnection, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a>               |
| stuittiti        | HRESULT | <a href="#">OleCreateLinkToFile(_In_ LPCOLESTR lpszFileName, _In_ REFIID riid, _In_ DWORD renderopt, _In_ LPFORMATETC lpFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)</a>                                                                                                                                                 |
|                  |         | <a href="#">OleCreateLinkToFileEx(_In_ LPCOLESTR lpszFileName, _In_ REFIID riid, _In_ DWORD</a>                                                                                                                                                                                                                                                                     |

|                 |          |                                                                                                                                                                                                                                                 |
|-----------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stuiuiuitttttti | HRESULT  | dwFlags, _In_ DWORD renderopt, _In_ ULONG cFormats, _In_ DWORD *rgAdvf, _In_ LPFORMATETC rgFormatEtc, _In_ IAdviseSink *lpAdviseSink, _Out_ DWORD *rgdwConnection, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj) |
| ttt             | HOLEMENU | OleCreateMenuDescriptor(_In_ HMENU hmenuCombined, _In_ LPOLEMENUGROUPWIDTHS lpMenuWidths)                                                                                                                                                       |
| ttuittti        | HRESULT  | OleCreateStaicFromData(_In_ LPDATAOBJECT pSrcDataObj, _In_ REFIID iid, _In_ DWORD renderopt, _In_ LPFORMATETC pFormatEtc, _In_ LPOLECLIENTSITE pClientSite, _In_ LPSTORAGE pStg, _Out_ LPVOID *ppvObj)                                          |
| ti              | VOID     | OleDestroyMenuDescriptor(_In_ HOLEMENU holemenu)                                                                                                                                                                                                |
| tti             | HRESULT  | OleDoAutoConvert(_In_ LPSTORAGE pStg, _Out_ LPCLSID pClsidNew)                                                                                                                                                                                  |
| tuitti          | HRESULT  | OleDraw(_In_ LPUNKNOWN pUnknown, _In_ DWORD dwAspect, _In_ HDC hdcDraw, _In_ LPCRECT lprcBounds)                                                                                                                                                |
| tuhuit          | HANDLE   | OleDuplicateData(_In_ HANDLE hSrc, _In_ CLIPFORMAT cfFormat, _In_ UINT uiFlags)                                                                                                                                                                 |
| i               | HRESULT  | OleFlushClipboard(void)                                                                                                                                                                                                                         |
| tti             | HRESULT  | OleGetAutoConvert(_In_ REFCLSID clsidOld, _Out_ LPCLSID pClsidNew)                                                                                                                                                                              |
| ti              | HRESULT  | OleGetClipboard(_Out_ LPDATAOBJECT *ppDataObj)                                                                                                                                                                                                  |
| tsit            | HGLOBAL  | OleGetIconOfClass(_In_ REFCLSID rclsid, _In_opt_ LPOLESTR lpszLabel, _In_ BOOL fUseTypeAsLabel)                                                                                                                                                 |
| sit             | HGLOBAL  | OleGetIconOfFile(_In_ LPOLESTR lpszPath, _In_ BOOL fUseFileAsLabel)                                                                                                                                                                             |
| ti              | HRESULT  | OleInitialize(_In_ LPVOID pvReserved)                                                                                                                                                                                                           |
| ti              | HRESULT  | OleIsCurrentClipboard(_In_ LPDATAOBJECT pDataObj)                                                                                                                                                                                               |
| ti              | BOOL     | OleIsRunning(_In_ LPOLEOBJECT pObject)                                                                                                                                                                                                          |
| tttti           | HRESULT  | OleLoad(_In_ LPSTORAGE pStg, _In_ REFIID riid, _In_ LPOLECLIENTSITE pClientSite, _Out_ LPVOID *ppvObj)                                                                                                                                          |
| ttti            | HRESULT  | OleLoadFromStream(_In_ LPSTREAM pStm, _In_ REFIID iidInterface, _Out_ LPVOID                                                                                                                                                                    |

|        |         |                                                                                                                                                                          |
|--------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        |         | *ppvObj)                                                                                                                                                                 |
| tiii   | HRESULT | OleLockRunning(_In_ LPUNKNOWN pUnknown, _In_ BOOL fLock, _In_ BOOL fLastUnlockCloses)                                                                                    |
| tssuit | HGLOBAL | OleMetafilePictFromIconAndLabel(_In_ HICON hIcon, _In_ LPOLESTR lpszLabel, _In_ LPOLESTR lpszSourceFile, _In_ UINT iIconIndex)                                           |
| tii    | HRESULT | OleNoteObjectVisible(_In_ LPUNKNOWN pUnknown, _In_ BOOL fVisible)                                                                                                        |
| ti     | HRESULT | OleQueryCreateFromData(_In_ LPDATAOBJECT pSrcDataObject)                                                                                                                 |
| ti     | HRESULT | OleQueryLinkFromData(_In_ LPDATAOBJECT pSrcDataObject)                                                                                                                   |
| tuiti  | HRESULT | OleRegEnumFormatEtc(_In_ REFCLSID clsid, _In_ DWORD dwDirection, _Out_ LPENUMFORMATETC *ppenum)                                                                          |
| tii    | HRESULT | OleRegEnumVerbs(_In_ REFCLSID clsid, _Out_ LPENUMOLEVERB *ppenum)                                                                                                        |
| tuiti  | HRESULT | OleRegGetMiscStatus(_In_ REFCLSID clsid, _In_ DWORD dwAspect, _Out_ DWORD *pdwStatus)                                                                                    |
| tuiti  | HRESULT | OleRegGetUserType(_In_ REFCLSID clsid, _In_ DWORD dwFormOfType, _Out_ LPOLESTR *pszUserType)                                                                             |
| ti     | HRESULT | OleRun(_In_ LPUNKNOWN pUnknown)                                                                                                                                          |
| tiii   | HRESULT | OleSave(_In_ LPPERSISTSTORAGE pPS, _In_ LPSTORAGE pStg, _In_ BOOL fSameAsLoad)                                                                                           |
| tii    | HRESULT | OleSaveToStream(_In_ LPPERSISTSTREAM pPStm, _In_ LPSTREAM pStm)                                                                                                          |
| tii    | HRESULT | OleSetAutoConverter(_In_ REFCLSID clsidOld, _In_ REFCLSID clsidNew)                                                                                                      |
| ti     | HRESULT | OleSetClipboard(_In_ LPDATAOBJECT pDataObj)                                                                                                                              |
| tii    | HRESULT | OleSetContainedObject(_In_ LPUNKNOWN pUnknown, _In_ BOOL fContained)                                                                                                     |
| tttiti | HRESULT | OleSetMenuDescriptor(_In_ HOLEMENU holemenu, _In_ HWND hwndFrame, _In_ HWND hwndActiveObject, _In_ LPOLEINPLACEFRAME lpFrame, _In_ LPOLEINPLACEACTIVEOBJECT lpActiveObj) |
|        |         | OleTranslateAccelerator(_In_                                                                                                                                             |

|             |                          |                                                                                                                                                                                                                             |
|-------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| titi        | HRESULT                  | LPOLEINPLACEFRAME lpFrame, _In_ LPOLEINPLACEFRAMEINFO lpFrameInfo, _In_ LPMSG lpmsg)                                                                                                                                        |
| i           | VOID                     | OleUninitialize(void)                                                                                                                                                                                                       |
| titi        | HRESULT                  | ProgIDFromCLSID(_In_ REFCLSID clsid, _Out_ LPOLESTR *lppszProgID)                                                                                                                                                           |
| stti        | HRESULT                  | PropSigNameToFmtid(_In_ const LPOLESTR pszName, _Out_ FMTID *pfmtid)                                                                                                                                                        |
| ti          | HRESULT                  | PropVariantClear(_Inout_ PROPVARIANT *pvar)                                                                                                                                                                                 |
| titi        | HRESULT                  | PropVariantCopy(_Out_ PROPVARIANT *pvarDest, _In_ const PROPVARIANT *pvarSrc)                                                                                                                                               |
| titi        | int                      | ReadClassSig(_In_ IStorage *pStg, _Out_ CLSID *pclsid)                                                                                                                                                                      |
| titi        | int                      | ReadClassStm(_In_ IStream *pStm, _Out_ CLSID *pclsid)                                                                                                                                                                       |
| titi        | int                      | ReadFmtUserTypeStg(_In_ IStorage *pStg, _Out_ CLIPFORMAT *pcf, _Out_ LPWSTR *lppszUserType)                                                                                                                                 |
| titi        | HRESULT                  | RegisterDragDrop(_In_ HWND hwnd, _In_ LPDROPTARGET pDropTarget)                                                                                                                                                             |
| ti          | VOID                     | ReleaseSigMedium(_In_ LPSTGMEDIUM pMedium)                                                                                                                                                                                  |
| ti          | HRESULT                  | RevokeDragDrop(_In_ HWND hwnd)                                                                                                                                                                                              |
| titi        | int                      | SetConvertSig(IStorage *pStg, BOOL fConvert)                                                                                                                                                                                |
| tuhttuc     | BOOLEAN                  | StgConvertPropertyToVariant(_In_ const SERIALIZEDPROPERTYVALUE *prop, _In_ USHORT CodePage, _Out_ PROPVARIANT *pvar, _In_ PMemoryAllocator *pma)                                                                            |
| tuhttuiuctt | SERIALIZEDPROPERTYVALUE* | StgConvertVariantToProperty(_In_ const PROPVARIANT *pvar, _In_ USHORT CodePage, _Out_opt_ SERIALIZEDPROPERTYVALUE *pprop, _Inout_ ULONG *pcb, _In_ PROPID pid, _Reserved_ BOOLEAN fReserved, _Inout_opt_ ULONG *pcIndirect) |
| wuiuiti     | HRESULT                  | StgCreateDocBk(_In_ const WCHAR *pwcsName, _In_ DWORD grfMode, _In_ DWORD reserved, _Out_ IStorage **ppstgOpen)                                                                                                             |
| tuiuiti     | HRESULT                  | StgCreateDocFileOnLockBytes(_In_ ILockBytes *plkbyt, _In_ DWORD grfMode,                                                                                                                                                    |

|            |         |                                                                                                                                                                                                                                                          |
|------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |         | _In_ DWORD reserved, _Out_ IStorage<br>**ppstgOpen)                                                                                                                                                                                                      |
| tuiti      | HRESULT | StgCreatePropSetStg(_In_ IStorage *pStorage,<br>DWORD dwReserved, _Out_<br>IPropertySetStorage **ppPropSetStg)                                                                                                                                           |
| tttuiti    | HRESULT | StgCreatePropStg(_In_ IUnknown *pUnk, _In_<br>REFFMID fmtid, _In_ const CLSID *pclsid,<br>_In_ DWORD grfFlags, _In_ DWORD<br>dwReserved, _Out_ IPropertyStorage<br>**ppPropStg)                                                                          |
| wuiuiittti | int     | StgCreateStorageEx(_In_ const WCHAR<br>*pwcsName, _In_ DWORD grfMode, _In_<br>STGFMT stgfmt, _In_ DWORD grfAttrs, _In_<br>STGOPTIONS *pStgOptions, _In_<br>PSECURITY_DESCRIPTOR<br>*pSecurityDescriptor, _In_ REFIID riid, _Out_<br>void **ppObjectOpen) |
| wti        | int     | StgGetFillLockBytesOnFile(_In_ OLECHAR<br>*pwcsName, _Out_ IFillLockBytes **ppflb)                                                                                                                                                                       |
| tti        | int     | StgGetFillLockBytesOnLockBytes(_In_<br>ILockBytes *pilb, _Out_ IFillLockBytes<br>**ppflb)                                                                                                                                                                |
| wi         | int     | StgIsStorageFile(_In_ const WCHAR<br>*pwcsName)                                                                                                                                                                                                          |
| ti         | int     | StgIsStorageLockBytes(ILockBytes *plkbyt)                                                                                                                                                                                                                |
| tuiti      | int     | StgOpenAsyncDecfileOnIFillLockBytes(_In_<br>IFillLockBytes *ppflb, _In_ DWORD grfmode,<br>_In_ DWORD asyncFlags, _Out_ IStorage<br>**ppstgOpen)                                                                                                          |
| tttuiti    | HRESULT | StgOpenPropStg(_In_ IUnknown *pUnk, _In_<br>REFFMID fmtid, _In_ DWORD grfFlags, _In_<br>DWORD dwReserved, _Out_ IPropertyStorage<br>**ppPropStg)                                                                                                         |
| wtuiti     | HRESULT | StgOpenStorage(_In_ const WCHAR<br>*pwcsName, _In_ IStorage *pstgPriority, _In_<br>DWORD grfMode, _In_ SNB snbExclude, _In_<br>DWORD reserved, _Out_ IStorage<br>**ppstgOpen)                                                                            |
| wuiuiittti | HRESULT | StgOpenStorageEx(_In_ const WCHAR<br>*pwcsName, _In_ DWORD grfMode, _In_<br>STGFMT stgfmt, _In_ DWORD grfAttrs,<br>_Inout_ STGOPTIONS *pStgOptions, _In_ void<br>*reserved2, _In_ REFIID riid, _Out_ void<br>**ppObjectOpen)                             |
|            |         | StgOpenStorageOnLockBytes(_In_ ILockBytes<br>*plkbyt, _In_ IStorage *pStgPriority, _In_                                                                                                                                                                  |

|           |         |                                                                                                                                           |
|-----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------|
| ttuituiti | HRESULT | DWORD grfMode, _In_ SNB snbExclude, _In_ DWORD reserved, _Out_ IStorage **ppstgOpen)                                                      |
| tuiuhucui | ULONG   | StgPropertyLengthAsVariant(_In_ const SERIALIZEDPROPERTYVALUE *pProp, _In_ ULONG cbProp, _In_ USHORT CodePage, _Reserved_ BYTE bReserved) |
| wttti     | int     | StgSetTimes(_In_ WCHAR const *lpszName, _In_ FILETIME const *pctime, _In_ FILETIME const *patime, _In_ FILETIME const *pmtime)            |
| tii       | HRESULT | StringFromCLSID(_In_ REFCLSID rclsid, _Out_ LPOLESTR *lplpsz)                                                                             |
| tsii      | int     | StringFromGUID2(_In_ REFGUID rguid, _Out_ LPOLESTR lpsz, _In_ int cchMax)                                                                 |
| tii       | HRESULT | StringFromIID(_In_ REFIID rclsid, _Out_ LPOLESTR *lplpsz)                                                                                 |
| tii       | int     | WriteClassStg(_In_ IStorage *pStg, _In_ REFCLSID rclsid)                                                                                  |
| tii       | int     | WriteClassStm(_In_ IStream *pStm, _In_ REFCLSID rclsid)                                                                                   |
| tuhti     | int     | WriteFmtUserTypeStg(_In_ IStorage *pStg, _In_ CLIPFORMAT cf, _In_ LPWSTR *lpszUserType)                                                   |

# Oleacc.dll

|          |         |                                                                                                                                                       |
|----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuitti | int     | AccessibleChildren(_In_ IAccessible *paccContainer, _In_ LONG iChildStart, _In_ LONG cChildren, _Out_ VARIANT *rgvarChildren, _Out_ LONG *pcObtained) |
| tuiuitti | int     | AccessibleObjectFromEvent(_In_ HWND hwnd, _In_ DWORD dwObjectID, _In_ DWORD dwChildID, _Out_ IAccessible **ppacc, _Out_ VARIANT *pvarChild)           |
| i6tti    | int     | AccessibleObjectFromPoint(_In_ POINT ptScreen, _Out_ IAccessible **ppacc, _Out_ VARIANT *pvarChild)                                                   |
| tuitti   | int     | AccessibleObjectFromWindow(_In_ HWND hwnd, _In_ DWORD dwObjectID, _In_ REFIID riid, _Out_ void **ppvObject)                                           |
| tuitti   | int     | CreateStdAccessibleObject(_In_ HWND hwnd, _In_ LONG idObject, _In_ REFIID riidInterface, _Out_ void **ppvObject)                                      |
| tsuitti  | int     | CreateStdAccessibleProxy(_In_ HWND hwnd, _In_ LPCTSTR pszClassName, _In_ LONG idObject, _In_ REFIID riidInterface, _Out_ void **ppvObject)            |
| tauitti  | int     | CreateStdAccessibleProxyA(_In_ HWND hwnd, _In_ LPCSTR pszClassName, _In_ LONG idObject, _In_ REFIID riidInterface, _Out_ void **ppvObject)            |
| twuitti  | int     | CreateStdAccessibleProxyW(_In_ HWND hwnd, _In_ LPCWSTR pszClassName, _In_ LONG idObject, _In_ REFIID riidInterface, _Out_ void **ppvObject)           |
| tti      | VOID    | GetOleaccVersionInfo(_Out_ DWORD *pdwVer, _Out_ DWORD *pdwBuild)                                                                                      |
| tt       | HANDLE  | GetProcessHandleFromHwnd(_In_ HWND hwnd)                                                                                                              |
| uisuiui  | UINT    | GetRoleTex(_In_ DWORD dwRole, _Out_ LPTSTR lpszRole, _In_ UINT cchRoleMax)                                                                            |
| uiauiui  | UINT    | GetRoleTextA(_In_ DWORD dwRole, _Out_ LPSTR lpszRole, _In_ UINT cchRoleMax)                                                                           |
| uiwuiui  | UINT    | GetRoleTextW(_In_ DWORD dwRole, _Out_ LPWSTR lpszRole, _In_ UINT cchRoleMax)                                                                          |
| uisuiui  | UINT    | GetStateTex(_In_ DWORD dwStateBit, _Out_ LPTSTR lpszStateBit, _In_ UINT cchStateBitMax)                                                               |
| uiauiui  | UINT    | GetStateTextA(_In_ DWORD dwStateBit, _Out_ LPSTR lpszStateBit, _In_ UINT cchStateBitMax)                                                              |
| uiwuiui  | UINT    | GetStateTextW(_In_ DWORD dwStateBit, _Out_ LPWSTR lpszStateBit, _In_ UINT cchStateBitMax)                                                             |
| tttt     | LRESULT | LresultFromObject(_In_ REFIID riid, _In_ WPARAM wParam, _In_ LPUNKNOWN pAcc)                                                                          |
| tttti    | int     | ObjectFromLresult(_In_ LRESULT lResult, _In_ REFIID riid, _In_ WPARAM wParam, _Out_ void **ppvObject)                                                 |

tti

int

[WindowFromAccessibleObject](#)(\_In\_ IAccessible \*pacc, \_Out\_ HWND \*phwnd)

# OleAut32.dll

|            |               |                                                                                                                                                               |
|------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tti        | VOID          | BSTR_UserFree(_In_ unsigned long *pFlags, _In_ BSTR *pBstr)                                                                                                   |
| tatt       | unsignedchar* | BSTR_UserMarshal(_In_ unsigned long *pFlags, _Inout_ unsigned char *pBuffer, _In_ BSTR *pBstr)                                                                |
| tuitui     | unsignedlong  | BSTR_UserSize(_In_ unsigned long *pFlags, _In_ unsigned long Offset, _In_ BSTR *pBstr)                                                                        |
| tatt       | unsignedchar* | BSTR_UserUnmarshal(_In_ unsigned long *pFlags, _In_ unsigned char *pBuffer, _Out_ BSTR *pBstr)                                                                |
| tti        | HRESULT       | BstrFromVector(_In_ SAFEARRAY *psa, _Out_ BSTR *pbstr)                                                                                                        |
| ti         | VOID          | ClearCustData(LPCUSTDATA pCustData)                                                                                                                           |
| tuiti      | HRESULT       | CreateDispTypeInfo(INTERFACEDATA *pidata, LCID lcid, ITypeInfo **pptinfo)                                                                                     |
| ti         | HRESULT       | CreateErrorInfo(_Out_ ICreateErrorInfo **pperrinfo)                                                                                                           |
| ttti       | HRESULT       | CreateStdDispatch(IUnknown *punkOuter, void *pvThis, ITypeInfo *ptinfo, IUnknown **ppunkStdDisp)                                                              |
| uisti      | HRESULT       | CreateTypeLib(SYSKIND syskind, LPCOLESTR szFile, ICreateTypeLib **ppctlib)                                                                                    |
| uisti      | HRESULT       | CreateTypeLib2(SYSKIND syskind, LPCOLESTR szFile, ICreateTypeLib2 **ppctlib)                                                                                  |
| ttiuhittti | HRESULT       | DispCallFunc(void *pvInstance, ULONG_PTR oVft, CALLCONV cc, VARTYPE vtReturn, UINT cActuals, VARTYPE *prgvt, VARIANTARG **prgpvarg, VARIANT *pvargResult)     |
| twuiti     | HRESULT       | DispGetIDsOfNames(ITypeInfo *ptinfo, _In_ OLECHAR **rgszNames, UINT cNames, _Out_ DISPID *rgdispid)                                                           |
| tuihitti   | HRESULT       | DispGetParam(_In_ DISPPARAMS *pdispparams, UINT position, VARTYPE vtTarg, _Out_ VARIANT *pvarResult, _Out_opt_ UINT *puArgErr)                                |
| ttiuhittti | HRESULT       | DispInvoke(void *_this, ITypeInfo *ptinfo, DISPID dispidMember, WORD wFlags, DISPPARAMS *pparams, VARIANT *pvarResult, EXCEPINFO *pexcepinfo, UINT *puArgErr) |
| uhuhti     | int           | DosDateTimeToVariantTime(_In_ USHORT wDosDate, _In_ USHORT wDosTime, _Out_ DOUBLE *pvtime)                                                                    |
| ttti       | HRESULT       | GetActiveObject(_In_ REFCLSID rclsid, _Reserved_ void *pvReserved, _Out_ IUnknown **ppunk)                                                                    |
| uiti       | HRESULT       | GetAltMonthNames(_In_ LCID lcid, _Out_ LPOLESTR **prgp)                                                                                                       |
| uiti       | HRESULT       | GetErrorInfo(_In_ ULONG dwReserved, _Out_ IErrorInfo **pperrinfo)                                                                                             |
|            |               | GetRecordInfoFromGuids(_In_ REFGUID rGuidTypeLib, _In_                                                                                                        |

|                    |               |                                                                                                                                                                  |
|--------------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiuitti         | HRESULT       | ULONG uVerMajor, _In_ ULONG uVerMinor, _In_ LCID lcid, _In_ REFGUID rGuidTypeInfo, _Out_ IRecordInfo **ppRecInfo)                                                |
| titi               | HRESULT       | GetRecordInfoFromTypeInfo(_In_ ITypeInfo *pTypeInfo, _Out_ IRecordInfo **ppRecInfo)                                                                              |
| ti                 | HRESULT       | GetVarConversionLocaleSetting(_Out_ ULONG *dwFlags)                                                                                                              |
| uiuiwui            | ULONG         | LHashValOfNameSys(SYSKIND syskind, LCID lcid, const OLECHAR *szName)                                                                                             |
| uiuiaui            | ULONG         | LHashValOfNameSysA(SYSKIND syskind, LCID lcid, LPCSTR szName)                                                                                                    |
| tuhuhuiti          | HRESULT       | LoadRegTypeLib(REFGUID rguid, WORD wVerMajor, WORD wVerMinor, LCID lcid, ITypeLib **pptlib)                                                                      |
| sti                | HRESULT       | LoadTypeLib(LPCOLESTR szFile, ITypeLib **pptlib)                                                                                                                 |
| suiti              | HRESULT       | LoadTypeLibEx(LPCOLESTR szFile, REGKIND regkind, ITypeLib **pptlib)                                                                                              |
| tattt              | unsignedchar* | LPSAFEARRAY_Marshal(_In_ unsigned long *pFlags, _Inout_ unsigned char *pBuffer, _In_ LPSAFEARRAY *ppSafeArray, _In_ const IID *piid)                             |
| tuitui             | unsignedlong  | LPSAFEARRAY_Size(_In_ unsigned long *pFlags, _In_ unsigned long Offset, _In_ LPSAFEARRAY *ppSafeArray, _In_ const IID *piid)                                     |
| tattt              | unsignedchar* | LPSAFEARRAY_Unmarshal(_In_ unsigned long *pFlags, _Inout_ unsigned char *pBuffer, _In_ LPSAFEARRAY *ppSafeArray, _In_ const IID *piid)                           |
| titi               | VOID          | LPSAFEARRAY_UserFree(_In_ unsigned long *pFlags, _In_ LPSAFEARRAY *ppSafeArray)                                                                                  |
| tatt               | unsignedchar* | LPSAFEARRAY_UserMarshal(_In_ unsigned long *pFlags, _Inout_ unsigned char *pBuffer, _In_ LPSAFEARRAY *ppSafeArray)                                               |
| tuitui             | unsignedlong  | LPSAFEARRAY_UserSize(_In_ unsigned long *pFlags, _In_ unsigned long Offset, _In_ LPSAFEARRAY *ppSafeArray)                                                       |
| tatt               | unsignedchar* | LPSAFEARRAY_UserUnmarshal(_In_ unsigned long *pFlags, _Inout_ unsigned char *pBuffer, _In_ LPSAFEARRAY *ppSafeArray)                                             |
| ui                 | ULONG         | OsBuildVersion(void)                                                                                                                                             |
| i                  | VOID          | OsEnablePerUserTLibRegistration(void)                                                                                                                            |
| titi               | HRESULT       | OleCreateFontIndirect(_In_ LPFONTDESC lpFontDesc, _In_ REFIID riid, _Out_ LPVOID *lplpvObj)                                                                      |
| titi               | HRESULT       | OleCreatePictureIndirect(_In_ LPPICTDESC lpPictDesc, _In_ REFIID riid, _In_ BOOL fOwn, _Out_ LPVOID *lplpvObj)                                                   |
| tuiuisuituituiuiti | HRESULT       | OleCreatePropertyFrame(_In_ HWND hwndOwner, _In_ UINT x, _In_ UINT y, _In_ LPCOLESTR lpszCaption, _In_ ULONG cObjects, _In_ LPUNKNOWN *ppUnk, _In_ ULONG cPages, |

|              |         |                                                                                                                                                                                                             |
|--------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |         | <code>_In_ LPCLSID pPageClsID, _In_ LCID lcid, _In_ DWORD dwReserved, _In_ LPVOID pvReserved)</code>                                                                                                        |
| ti           | HRESULT | <code>OleCreatePropertyFrameIndirect(_In_ LPOCPFIPARAMS lpParams)</code>                                                                                                                                    |
| ttt          | HCURSOR | <code>OleIconToCursor(_In_ HINSTANCE hinstExe, _In_ HICON hIcon)</code>                                                                                                                                     |
| tuiitti      | HRESULT | <code>OleLoadPicture(_In_ LPSTREAM lpstream, _In_ LONG lSize, _In_ BOOL fRunmode, _In_ REFIID riid, _Out_ LPVOID *lpIplvObj)</code>                                                                         |
| tuiituiiuiti | HRESULT | <code>OleLoadPictureEx(_In_ LPSTREAM lpstream, _In_ LONG lSize, _In_ BOOL fRunmode, _In_ REFIID riid, _In_ DWORD xSizeDesired, _In_ DWORD ySizeDesired, _In_ DWORD dwFlags, _Out_ LPVOID *lpIplvObj)</code> |
| tti          | HRESULT | <code>OleLoadPictureFile(_In_ VARIANT varFileName, _Out_ LPDISPATCH *lpIpldispPicture)</code>                                                                                                               |
| tuiuiiuiti   | HRESULT | <code>OleLoadPictureFileEx(_In_ VARIANT varFileName, _In_ DWORD xSizeDesired, _In_ DWORD ySizeDesired, _In_ DWORD dwFlags, _Out_ LPDISPATCH *lpIpldispPicture)</code>                                       |
| stuiiitti    | HRESULT | <code>OleLoadPicturePath(_In_ LPOLESTR szURLorPath, _In_ LPUNKNOWN punkCaller, _In_ DWORD dwReserved, _In_ OLE_COLOR clrReserved, _In_ REFIID riid, _Out_ LPVOID *ppvRet)</code>                            |
| tsi          | HRESULT | <code>OleSavePictureFile(_In_ LPDISPATCH lpdispPicture, _In_ BSTR bstrFileName)</code>                                                                                                                      |
| uitti        | HRESULT | <code>OleTranslateColor(_In_ OLE_COLOR clr, _In_ HPALETTE hpal, _Out_ COLORREF *lpcolorref)</code>                                                                                                          |
| tuhuhuisi    | HRESULT | <code>QueryPathOfRegTypeLib(REFGUID guid, USHORT wMaj, USHORT wMin, LCID lcid, _Out_ LPBSTR lpbstrPathName)</code>                                                                                          |
| ttuiti       | HRESULT | <code>RegisterActiveObject(IUnknown *punk, REFCLSID rclsid, DWORD dwFlags, DWORD *pdwRegister)</code>                                                                                                       |
| tssi         | HRESULT | <code>RegisterTypeLib(ITypeLib *ptlib, _In_ LPCOLESTR szFullPath, _In_opt_ LPCOLESTR szHelpDir)</code>                                                                                                      |
| twwi         | HRESULT | <code>RegisterTypeLibForUser(ITypeLib *ptlib, _In_ OLECHAR *szFullPath, _In_opt_ OLECHAR *szHelpDir)</code>                                                                                                 |
| uiti         | HRESULT | <code>RevokeActiveObject(_In_ DWORD dwRegister, _Reserved_ void *pvReserved)</code>                                                                                                                         |
| tti          | HRESULT | <code>SafeArrayAccessData(_In_ SAFEARRAY *psa, _Out_ void **ppvData)</code>                                                                                                                                 |
| ti           | HRESULT | <code>SafeArrayAllocData(_In_ SAFEARRAY *psa)</code>                                                                                                                                                        |
| uiti         | HRESULT | <code>SafeArrayAllocDescriptor(_In_ UINT cDims, _Out_ SAFEARRAY **ppsaOut)</code>                                                                                                                           |
| uhuiti       | HRESULT | <code>SafeArrayAllocDescriptorEx(_In_ VARTYPE vt, _In_ UINT</code>                                                                                                                                          |

|          |            |                                                                                                         |
|----------|------------|---------------------------------------------------------------------------------------------------------|
|          |            | cDims, _Out_ SAFEARRAY **ppsaOut)                                                                       |
| tii      | HRESULT    | SafeArrayCopy(_In_ SAFEARRAY *psa, _Out_ SAFEARRAY **ppsaOut)                                           |
| tii      | HRESULT    | SafeArrayCopyData(_In_ SAFEARRAY *psaSource, _In_ SAFEARRAY *psaTarget)                                 |
| uhuiitt  | SAFEARRAY* | SafeArrayCreate(_In_ VARTYPE vt, _In_ UINT cDims, _In_ SAFEARRAYBOUND *rgsabound)                       |
| uhuiitt  | SAFEARRAY* | SafeArrayCreateEx(_In_ VARTYPE vt, _In_ UINT cDims, _In_ SAFEARRAYBOUND *rgsabound, _In_ PVOID pvExtra) |
| uhuiiuit | SAFEARRAY* | SafeArrayCreateVector(_In_ VARTYPE vt, _In_ LONG lLbound, _In_ ULONG cElements)                         |
| uhuiiuit | SAFEARRAY* | SafeArrayCreateVectorEx(_In_ VARTYPE vt, _In_ LONG lLbound, _In_ ULONG cElements, _In_ PVOID pvExtra)   |
| ti       | HRESULT    | SafeArrayDestroy(_In_ SAFEARRAY *psa)                                                                   |
| ti       | HRESULT    | SafeArrayDestroyData(_In_ SAFEARRAY *psa)                                                               |
| ti       | HRESULT    | SafeArrayDestroyDescriptor(_In_ SAFEARRAY *psa)                                                         |
| tui      | UINT       | SafeArrayGetDim(_In_ SAFEARRAY *psa)                                                                    |
| tii      | HRESULT    | SafeArrayGetElement(_In_ SAFEARRAY *psa, _In_ LONG *rgIndices, _Out_ void *pv)                          |
| tui      | UINT       | SafeArrayGetElementSize(_In_ SAFEARRAY *psa)                                                            |
| tii      | HRESULT    | SafeArrayGetIID(_In_ SAFEARRAY *psa, _Out_ GUID *pguid)                                                 |
| tuiti    | HRESULT    | SafeArrayGetLBound(_In_ SAFEARRAY *psa, _In_ UINT nDim, _Out_ LONG *plLbound)                           |
| tii      | HRESULT    | SafeArrayGetRecordInfo(_In_ SAFEARRAY *psa, _Out_ IRecordInfo **prinfo)                                 |
| tuiti    | HRESULT    | SafeArrayGetUBound(_In_ SAFEARRAY *psa, _In_ UINT nDim, _Out_ LONG *plUbound)                           |
| tii      | HRESULT    | SafeArrayGetVartype(_In_ SAFEARRAY *psa, _Out_ VARTYPE *pvt)                                            |
| ti       | HRESULT    | SafeArrayLock(_In_ SAFEARRAY *psa)                                                                      |
| tii      | HRESULT    | SafeArrayPutElement(_In_ SAFEARRAY *psa, _In_ LONG *rgIndices, _Out_ void **ppvData)                    |
| tii      | HRESULT    | SafeArrayPutElement(_In_ SAFEARRAY *psa, _In_ LONG *rgIndices, _In_ void *pv)                           |
| tii      | HRESULT    | SafeArrayRedim(_Inout_ SAFEARRAY *psa, _In_ SAFEARRAYBOUND *psaboundNew)                                |
| tii      | HRESULT    | SafeArraySetIID(_In_ SAFEARRAY *psa, _In_ REFGUID guid)                                                 |
| tii      | HRESULT    | SafeArraySetRecordInfo(_In_ SAFEARRAY *psa, _In_ IRecordInfo *prinfo)                                   |
| ti       | HRESULT    | SafeArrayUnaccessData(_In_ SAFEARRAY *psa)                                                              |

|            |         |                                                                                                           |
|------------|---------|-----------------------------------------------------------------------------------------------------------|
| ti         | HRESULT | SafeArrayUnlock(_In_ SAFEARRAY *psa)                                                                      |
| uiti       | HRESULT | SetErrorInfo(_In_ ULONG dwReserved, _In_opt_ IErrorInfo *perrinfo)                                        |
| i          | VOID    | SetOaNoCache(void)                                                                                        |
| uii        | HRESULT | SetVarConversionLocaleSetting(_In_ ULONG dwFlags)                                                         |
| ws         | BSTR    | SysAllocString(_In_opt_ const OLECHAR *psz)                                                               |
| auis       | BSTR    | SysAllocStringByteLen(_In_opt_ LPCWSTR psz, _In_ UINT len)                                                |
| wuis       | BSTR    | SysAllocStringLen(_In_ const OLECHAR *strIn, _In_ UINT ui)                                                |
| si         | VOID    | SysFreeString(_In_opt_ BSTR bstrString)                                                                   |
| twi        | int     | SysReAllocString(_Inout_ BSTR *pbstr, _In_opt_ const OLECHAR *psz)                                        |
| twuii      | int     | SysReAllocStringLen(_Inout_ BSTR *pbstr, _In_opt_ const OLECHAR *psz, _In_ unsigned int len)              |
| sui        | UINT    | SysStringByteLen(_In_opt_ BSTR bstr)                                                                      |
| sui        | UINT    | SysStringLen(_In_opt_ BSTR bstr)                                                                          |
| tii        | int     | SystemTimeToVariantTime(_In_ LPSYSTEMTIME lpSystemTime, _Out_ DOUBLE *pvtime)                             |
| tuhuhuiiii | HRESULT | UnRegisterTypeLib(REFGUID libID, WORD wVerMajor, WORD wVerMinor, LCID lcid, SYSKIND syskind)              |
| tuhuhuiiii | HRESULT | UnRegisterTypeLibForUser(REFGUID libID, WORD wMajorVerNum, WORD wMinorVerNum, LCID lcid, SYSKIND syskind) |
| tii        | HRESULT | VarAbs(_In_ LPVARIANT pvarIn, _Out_ LPVARIANT pvarResult)                                                 |
| tiii       | HRESULT | VarAdd(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                     |
| tiii       | HRESULT | VarAnd(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                     |
| i6ti       | HRESULT | VarBoolFromCy(_In_ CY cyIn, _Out_ VARIANT_BOOL *pboolOut)                                                 |
| dii        | HRESULT | VarBoolFromDate(_In_ DATE dateIn, _Out_ VARIANT_BOOL *pboolOut)                                           |
| tii        | HRESULT | VarBoolFromDec(_In_ const DECIMAL *pdecIn, _Out_ VARIANT_BOOL *pboolOut)                                  |
| tuiti      | HRESULT | VarBoolFromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ VARIANT_BOOL *pboolOut)                         |
| cti        | HRESULT | VarBoolFromI1(_In_ CHAR cIn, _Out_ VARIANT_BOOL *pboolOut)                                                |
| htii       | HRESULT | VarBoolFromI2(_In_ SHORT sIn, _Out_ VARIANT_BOOL *pboolOut)                                               |

|              |         |                                                                                                        |
|--------------|---------|--------------------------------------------------------------------------------------------------------|
| uiti         | HRESULT | VarBoolFromI4(_In_ LONG lIn, _Out_ VARIANT_BOOL *pboolOut)                                             |
| i6ti         | HRESULT | VarBoolFromI8(_In_ LONG64 i64In, _Out_ VARIANT_BOOL *pboolOut)                                         |
| fti          | HRESULT | VarBoolFromR4(_In_ FLOAT fltIn, _Out_ VARIANT_BOOL *pboolOut)                                          |
| dti          | HRESULT | VarBoolFromR8(_In_ DOUBLE dblIn, _Out_ VARIANT_BOOL *pboolOut)                                         |
| suiuiti      | HRESULT | VarBoolFromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ VARIANT_BOOL *pboolOut) |
| ucti         | HRESULT | VarBoolFromUI1(_In_ BYTE bIn, _Out_ VARIANT_BOOL *pboolOut)                                            |
| uhti         | HRESULT | VarBoolFromUI2(_In_ USHORT uiIn, _Out_ VARIANT_BOOL *pboolOut)                                         |
| uiti         | HRESULT | VarBoolFromUI4(_In_ ULONG ulIn, _Out_ VARIANT_BOOL *pboolOut)                                          |
| ui6ti        | HRESULT | VarBoolFromUI8(_In_ ULONG64 i64In, _Out_ VARIANT_BOOL *pboolOut)                                       |
| sssi         | HRESULT | VarBstrCat(_In_ BSTR bstrLeft, _In_ BSTR bstrRight, _Out_ LPBSTR pbstrResult)                          |
| ssuiiii      | HRESULT | VarBstrCmp(_In_ BSTR bstrLeft, _In_ BSTR bstrRight, _In_ LCID lcid, _In_ ULONG dwFlags)                |
| huiuiti      | HRESULT | VarBstrFromBool(_In_ VARIANT_BOOL boolIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)    |
| i6uiiuiti    | HRESULT | VarBstrFromC(_In_ CY cyIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)                   |
| duiuiti      | HRESULT | VarBstrFromDate(_In_ DATE dateIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)            |
| tuiuiti      | HRESULT | VarBstrFromDec(_In_ const DECIMAL *pdecIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)   |
| tuiuiti      | HRESULT | VarBstrFromDisp(IDispatch *pdispIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)          |
| cuiiuiti     | HRESULT | VarBstrFromI1(_In_ CHAR cIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)                 |
| huiiuiti     | HRESULT | VarBstrFromI2(_In_ SHORT iVal, _In_ LCID lcid, _In_ ULONG dwFlags, BSTR *pbstrOut)                     |
| uiiuiiuiti   | HRESULT | VarBstrFromI4(_In_ LONG lIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)                 |
| i6uiiuiiuiti | HRESULT | VarBstrFromI8(_In_ LONG64 i64In, _In_ LCID lcid, _In_ unsigned long dwFlags, _Out_ BSTR *pbstrOut)     |
|              |         | VarBstrFromR4(_In_ FLOAT fltIn, _In_ LCID lcid, _In_ ULONG                                             |

|           |         |                                                                                                       |
|-----------|---------|-------------------------------------------------------------------------------------------------------|
| fuiiiti   | HRESULT | dwFlags, _Out_ BSTR *pbstrOut)                                                                        |
| duiuiiti  | HRESULT | VarBstrFromR8(_In_ DOUBLE dblIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)            |
| ucuiiiti  | HRESULT | VarBstrFromUI1(_In_ BYTE bVal, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)              |
| uhuiiiti  | HRESULT | VarBstrFromUI2(_In_ USHORT uiIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)            |
| uiuiiiti  | HRESULT | VarBstrFromUI4(_In_ ULONG ulIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)             |
| ui6uiiiti | HRESULT | VarBstrFromUI8(_In_ ULONG64 ui64In, _In_ LCID lcid, _In_ unsigned long dwFlags, _Out_ BSTR *pbstrOut) |
| ttti      | HRESULT | VarCat(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                 |
| ttuiiiti  | HRESULT | VarCmp(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _In_ LCID lcid, _In_ ULONG dwFlags)         |
| i6ti      | HRESULT | VarCyAbs(_In_ CY cyIn, _Out_ LPCY pcyResult)                                                          |
| i6i6ti    | HRESULT | VarCyAdd(_In_ CY cyLeft, _In_ CY cyRight, _Out_ LPCY pcyResult)                                       |
| i6i6i     | HRESULT | VarCyCmp(_In_ CY cyLeft, _In_ CY cyRight)                                                             |
| i6di      | HRESULT | VarCyCmpR8(_In_ CY cyLeft, _In_ double dblRight)                                                      |
| i6ti      | HRESULT | VarCyFix(_In_ CY cyIn, _Out_ LPCY pcyResult)                                                          |
| hti       | HRESULT | VarCyFromBool(_In_ VARIANT_BOOL boolIn, _Out_ CY *pcyOut)                                             |
| diti      | HRESULT | VarCyFromDate(_In_ DATE dateIn, _Out_ CY *pcyOut)                                                     |
| tti       | HRESULT | VarCyFromDec(_In_ const DECIMAL *pdecIn, _Out_ CY *pcyOut)                                            |
| tuiti     | HRESULT | VarCyFromDisp(_In_ IDispatch *pdispIn, _In_ LCID lcid, _Out_ CY *pcyOut)                              |
| cti       | HRESULT | VarCyFromI1(_In_ CHAR cIn, _Out_ CY *pcyOut)                                                          |
| hti       | HRESULT | VarCyFromI2(_In_ SHORT sIn, _Out_ CY *pcyOut)                                                         |
| uiti      | HRESULT | VarCyFromI4(_In_ LONG lIn, _Out_ CY *pcyOut)                                                          |
| i6ti      | HRESULT | VarCyFromI8(_In_ LONG64 i64In, _Out_ CY *pcyOut)                                                      |
| fti       | HRESULT | VarCyFromR4(_In_ FLOAT fltIn, _Out_ CY *pcyOut)                                                       |
| diti      | HRESULT | VarCyFromR8(_In_ DOUBLE dblIn, _Out_ CY *pcyOut)                                                      |
| suiiiti   | HRESULT | VarCyFromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ CY *pcyOut)              |
| ucti      | HRESULT | VarCyFromUI1(_In_ BYTE bIn, _Out_ CY *pcyOut)                                                         |
| uhti      | HRESULT | VarCyFromUI2(_In_ USHORT uiIn, _Out_ CY *pcyOut)                                                      |
|           |         |                                                                                                       |

|         |         |                                                                                                    |
|---------|---------|----------------------------------------------------------------------------------------------------|
| uiti    | HRESULT | VarCyFromUI4(_In_ ULONG ulIn, _Out_ CY *pcyOut)                                                    |
| ui6ti   | HRESULT | VarCyFromUI8(_In_ ULONG64 ui64In, _Out_ CY *pcyOut)                                                |
| i6ti    | HRESULT | VarCyInt(_In_ CY cyIn, _Out_ LPCY pcyResult)                                                       |
| i6i6ti  | HRESULT | VarCyMul(_In_ CY cyLeft, _In_ CY cyRight, _Out_ LPCY pcyResult)                                    |
| i6uiti  | HRESULT | VarCyMulI4(_In_ CY cyLeft, _In_ long lRight, _Out_ LPCY pcyResult)                                 |
| i6i6ti  | HRESULT | VarCyMulI8(_In_ CY cyLeft, _In_ LONG64 lRight, _Out_ LPCY pcyResult)                               |
| i6ti    | HRESULT | VarCyNeg(_In_ CY cyIn, _Out_ LPCY pcyResult)                                                       |
| i6iti   | HRESULT | VarCyRound(_In_ CY cyIn, _In_ int cDecimals, _Out_ LPCY pcyResult)                                 |
| i6i6ti  | HRESULT | VarCySub(_In_ CY cyLeft, _In_ CY cyRight, _Out_ LPCY pcyResult)                                    |
| hti     | HRESULT | VarDateFromBool(_In_ VARIANT_BOOL boolIn, _Out_ DATE *pdateOut)                                    |
| i6ti    | HRESULT | VarDateFromCy(_In_ CY cyIn, _Out_ DATE *pdateOut)                                                  |
| titi    | HRESULT | VarDateFromDec(_In_ const DECIMAL *pdecIn, _Out_ DATE *pdateOut)                                   |
| tuiti   | HRESULT | VarDateFromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ DATE *pdateOut)                          |
| cti     | HRESULT | VarDateFromI1(_In_ CHAR cIn, _Out_ DATE *pdateOut)                                                 |
| hti     | HRESULT | VarDateFromI2(_In_ SHORT sIn, _Out_ DATE *pdateOut)                                                |
| uiti    | HRESULT | VarDateFromI4(_In_ LONG lIn, _Out_ DATE *pdateOut)                                                 |
| i6ti    | HRESULT | VarDateFromI8(_In_ LONG64 i64In, _Out_ DATE *pdateOut)                                             |
| fti     | HRESULT | VarDateFromR4(_In_ FLOAT fltIn, _Out_ DATE *pdateOut)                                              |
| diti    | HRESULT | VarDateFromR8(_In_ DOUBLE dblIn, _Out_ DATE *pdateOut)                                             |
| suiuiti | HRESULT | VarDateFromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ DATE *pdateOut)     |
| tuiti   | HRESULT | VarDateFromUpdate(_In_ UDATE *pdateIn, _In_ ULONG dwFlags, _Out_ DATE *pdateOut)                   |
| tuiuiti | HRESULT | VarDateFromUpdateEx(_In_ UDATE *pdateIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ DATE *pdateOut) |
| ucti    | HRESULT | VarDateFromUI1(_In_ BYTE bIn, _Out_ DATE *pdateOut)                                                |
| uhti    | HRESULT | VarDateFromUI2(_In_ USHORT uiIn, _Out_ DATE *pdateOut)                                             |
| uiti    | HRESULT | VarDateFromUI4(_In_ ULONG ulIn, _Out_ DATE *pdateOut)                                              |
| ui6ti   | HRESULT | VarDateFromUI8(_In_ ULONG64 ui64In, _Out_ DATE *pdateOut)                                          |
|         |         | VarDscAbs(_In_ LPDECIMAL pdecIn, _Out_ LPDECIMAL                                                   |

|       |         |                                                                                                 |
|-------|---------|-------------------------------------------------------------------------------------------------|
| ttd   | HRESULT | pdecResult)                                                                                     |
| ttti  | HRESULT | VarDecAdd(_In_ LPDECIMAL pdecLeft, _In_ LPDECIMAL pdecRight, _Out_ LPDECIMAL pdecResult)        |
| ttd   | HRESULT | VarDecCmp(_In_ LPDECIMAL pdecLeft, _In_ LPDECIMAL pdecRight)                                    |
| tdi   | HRESULT | VarDecCmpR8(_In_ LPDECIMAL pdecLeft, _In_ double dblRight)                                      |
| ttti  | HRESULT | VarDecDiv(_In_ LPDECIMAL pdecLeft, _In_ LPDECIMAL pdecRight, _Out_ LPDECIMAL pdecResult)        |
| ttd   | HRESULT | VarDecFix(_In_ LPDECIMAL pdecIn, _Out_ LPDECIMAL pdecResult)                                    |
| hdi   | HRESULT | VarDecFromBool(_In_ VARIANT_BOOL boolIn, _Out_ DECIMAL *pdecOut)                                |
| i6td  | HRESULT | VarDecFromCy(_In_ CY cyIn, _Out_ DECIMAL *pdecOut)                                              |
| tdi   | HRESULT | VarDecFromDate(_In_ DATE dateIn, _Out_ DECIMAL *pdecOut)                                        |
| ttdi  | HRESULT | VarDecFromDisp(_In_ IDispatch *pdispIn, _In_ LCID lcid, _Out_ DECIMAL *pdecOut)                 |
| tdi   | HRESULT | VarDecFromI1(_In_ CHAR cIn, _Out_ DECIMAL *pdecOut)                                             |
| hdi   | HRESULT | VarDecFromI2(_In_ SHORT uiIn, _Out_ DECIMAL *pdecOut)                                           |
| tdi   | HRESULT | VarDecFromI4(_In_ LONG lIn, _Out_ DECIMAL *pdecOut)                                             |
| i6td  | HRESULT | VarDecFromI8(_In_ LONG64 i64In, _Out_ DECIMAL *pdecOut)                                         |
| fdi   | HRESULT | VarDecFromR4(_In_ FLOAT fltIn, _Out_ DECIMAL *pdecOut)                                          |
| tdi   | HRESULT | VarDecFromR8(_In_ DOUBLE dblIn, _Out_ DECIMAL *pdecOut)                                         |
| suidi | HRESULT | VarDecFromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ DECIMAL *pdecOut) |
| tdi   | HRESULT | VarDecFromUI1(_In_ BYTE bIn, _Out_ DECIMAL *pdecOut)                                            |
| hdi   | HRESULT | VarDecFromUI2(_In_ USHORT uiIn, _Out_ DECIMAL *pdecOut)                                         |
| tdi   | HRESULT | VarDecFromUI4(_In_ ULONG ulIn, _Out_ DECIMAL *pdecOut)                                          |
| ui6td | HRESULT | VarDecFromUI8(_In_ ULONG64 ui64In, _Out_ DECIMAL *pdecOut)                                      |
| ttd   | HRESULT | VarDecIn(_In_ LPDECIMAL pdecIn, _Out_ LPDECIMAL pdecResult)                                     |
| ttti  | HRESULT | VarDecMul(_In_ LPDECIMAL pdecLeft, _In_ LPDECIMAL pdecRight, _Out_ LPDECIMAL pdecResult)        |
| ttd   | HRESULT | VarDecNeg(_In_ LPDECIMAL pdecIn, _Out_ LPDECIMAL pdecResult)                                    |
|       |         | VarDecRound(_In_ LPDECIMAL pdecIn, _In_ int cDecimals,                                          |

|          |         |                                                                                                                                                               |
|----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| titi     | HRESULT | _Out_ LPDECIMAL pdecResult)                                                                                                                                   |
| ttti     | HRESULT | VarDecSub(_In_ LPDECIMAL pdecLeft, _In_ LPDECIMAL pdecRight, _Out_ LPDECIMAL pdecResult)                                                                      |
| ttti     | HRESULT | VarDiv(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                                                         |
| ttti     | HRESULT | VarExp(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                                                         |
| tti      | HRESULT | VarFix(_In_ LPVARIANT pvarIn, _Out_ LPVARIANT pvarResult)                                                                                                     |
| tsiuiti  | HRESULT | VarFormat(_In_ LPVARIANT pvarIn, _In_opt_ LPOLESTR pstrFormat, _In_ int iFirstDay, _In_ int iFirstWeek, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)             |
| tiiiuiti | HRESULT | VarFormatCurrency(_In_ LPVARIANT pvarIn, _In_ int iNumDig, _In_ int iIncLead, _In_ int iUseParens, _In_ int iGroup, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut) |
| tiuiti   | HRESULT | VarFormatDateTime(_In_ LPVARIANT pvarIn, _In_ int iNamedFormat, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)                                                     |
| tstuitui | HRESULT | VarFormatFromToken(_In_ LPVARIANT pvarIn, _In_opt_ LPOLESTR pstrFormat, _In_ LPBYTE pbTokCur, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut, _In_ LCID lcid)       |
| tiiiuiti | HRESULT | VarFormatNumber(_In_ LPVARIANT pvarIn, _In_ int iNumDig, _In_ int iIncLead, _In_ int iUseParens, _In_ int iGroup, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)   |
| tiiiuiti | HRESULT | VarFormatPercent(_In_ LPVARIANT pvarIn, _In_ int iNumDig, _In_ int iIncLead, _In_ int iUseParens, _In_ int iGroup, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)  |
| hai      | HRESULT | VarI1FromBool(_In_ VARIANT_BOOL boolIn, _Out_ CHAR *pcOut)                                                                                                    |
| i6ai     | HRESULT | VarI1FromCy(_In_ CY cyIn, _Out_ CHAR *pcOut)                                                                                                                  |
| dai      | HRESULT | VarI1FromDate(_In_ DATE dateIn, _Out_ CHAR *pcOut)                                                                                                            |
| tai      | HRESULT | VarI1FromDec(_In_ const DECIMAL *pdecIn, _Out_ CHAR *pcOut)                                                                                                   |
| tui      | HRESULT | VarI1FromDisp(_In_ IDispatch *pdispIn, _In_ LCID lcid, _Out_ CHAR *pcOut)                                                                                     |
| hai      | HRESULT | VarI1FromI2(_In_ SHORT uiIn, _Out_ CHAR *pcOut)                                                                                                               |
| ui       | HRESULT | VarI1FromI4(_In_ LONG lIn, _Out_ CHAR *pcOut)                                                                                                                 |
| i6ai     | HRESULT | VarI1FromI8(_In_ LONG64 i64In, _Out_ CHAR *pcOut)                                                                                                             |
| fai      | HRESULT | VarI1FromR4(_In_ FLOAT fltIn, _Out_ CHAR *pcOut)                                                                                                              |
| dai      | HRESULT | VarI1FromR8(_In_ DOUBLE dblIn, _Out_ CHAR *pcOut)                                                                                                             |
| sui      | HRESULT | VarI1FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ CHAR *pcOut)                                                                     |

|         |         |                                                                                            |
|---------|---------|--------------------------------------------------------------------------------------------|
| ucaai   | HRESULT | Var11FromUI1(_In_ BYTE bIn, _Out_ CHAR *pcOut)                                             |
| uhai    | HRESULT | Var11FromUI2(_In_ USHORT uiIn, _Out_ CHAR *pcOut)                                          |
| uiiai   | HRESULT | Var11FromUI4(_In_ ULONG ulIn, _Out_ CHAR *pcOut)                                           |
| ui6ai   | HRESULT | Var11FromUI8(_In_ ULONG64 i64In, _Out_ CHAR *pcOut)                                        |
| hti     | HRESULT | Var12FromBool(_In_ VARIANT_BOOL boolIn, _Out_ SHORT *psOut)                                |
| i6ti    | HRESULT | Var12FromCy(_In_ CY cyIn, SHORT *psOut)                                                    |
| dti     | HRESULT | Var12FromDate(_In_ DATE dateIn, _Out_ SHORT *psOut)                                        |
| titi    | HRESULT | Var12FromDec(_In_ const DECIMAL *pdecIn, _Out_ SHORT *psOut)                               |
| tuiti   | HRESULT | Var12FromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ SHORT *psOut)                      |
| cti     | HRESULT | Var12FromI1(_In_ CHAR cIn, _Out_ SHORT *psOut)                                             |
| uiti    | HRESULT | Var12FromI4(_In_ LONG lIn, _Out_ SHORT *psOut)                                             |
| i6ti    | HRESULT | Var12FromI8(_In_ LONG64 i64In, _Out_ SHORT *psOut)                                         |
| fti     | HRESULT | Var12FromR4(_In_ FLOAT fltIn, _Out_ SHORT *psOut)                                          |
| diti    | HRESULT | Var12FromR8(_In_ DOUBLE dblIn, _Out_ SHORT *psOut)                                         |
| suiuiti | HRESULT | Var12FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ SHORT *psOut) |
| ucti    | HRESULT | Var12FromUI1(_In_ BYTE bIn, _Out_ SHORT *psOut)                                            |
| uhti    | HRESULT | Var12FromUI2(_In_ USHORT uiIn, _Out_ SHORT *psOut)                                         |
| uiti    | HRESULT | Var12FromUI4(_In_ ULONG ulIn, _Out_ SHORT *psOut)                                          |
| ui6ti   | HRESULT | Var12FromUI8(_In_ ULONG64 ui64In, _Out_ SHORT *psOut)                                      |
| hti     | HRESULT | Var14FromBool(_In_ VARIANT_BOOL boolIn, _Out_ LONG *plOut)                                 |
| i6ti    | HRESULT | Var14FromCy(_In_ CY cyIn, _Out_ LONG *plOut)                                               |
| diti    | HRESULT | Var14FromDate(_In_ DATE dateIn, _Out_ LONG *plOut)                                         |
| titi    | HRESULT | Var14FromDec(_In_ const DECIMAL *pdecIn, _Out_ LONG *plOut)                                |
| tuiti   | HRESULT | Var14FromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ LONG *plOut)                       |
| cti     | HRESULT | Var14FromI1(_In_ CHAR cIn, _Out_ LONG *plOut)                                              |
| hti     | HRESULT | Var14FromI2(_In_ SHORT sIn, _Out_ LONG *plOut)                                             |
| i6ti    | HRESULT | Var14FromI8(_In_ LONG64 i64In, _Out_ LONG *plOut)                                          |
| fti     | HRESULT | Var14FromR4(_In_ FLOAT fltIn, _Out_ LONG *plOut)                                           |
| diti    | HRESULT | Var14FromR8(_In_ DOUBLE dblIn, _Out_ LONG *plOut)                                          |
|         |         | Var14FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_                                    |

|         |               |                                                                                                                                       |
|---------|---------------|---------------------------------------------------------------------------------------------------------------------------------------|
| suiuiti | HRESULT       | ULONG dwFlags, _Out_ LONG *plOut)                                                                                                     |
| ucti    | HRESULT       | Var14FromUI1(_In_ BYTE bIn, _Out_ LONG *plOut)                                                                                        |
| uhti    | HRESULT       | Var14FromUI2(_In_ USHORT uiIn, _Out_ LONG *plOut)                                                                                     |
| uiti    | HRESULT       | Var14FromUI4(_In_ ULONG ulIn, _Out_ LONG *plOut)                                                                                      |
| ui6ti   | HRESULT       | Var14FromUI8(_In_ ULONG64 ui64In, _Out_ LONG *plOut)                                                                                  |
| hti     | HRESULT       | Var18FromBool(_In_ VARIANT_BOOL boolIn, _Out_ LONG64 *pi64Out)                                                                        |
| i6ti    | HRESULT       | Var18FromCy(_In_ CY cyIn, _Out_ LONG64 *pi64Out)                                                                                      |
| dti     | HRESULT       | Var18FromDate(_In_ DATE dateIn, _Out_ LONG64 *pi64Out)                                                                                |
| titi    | HRESULT       | Var18FromDec(_In_ const DECIMAL *pdecIn, _Out_ LONG64 *pi64Out)                                                                       |
| tuiti   | HRESULT       | Var18FromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ LONG64 *pi64Out)                                                              |
| cti     | HRESULT       | Var18FromI1(_In_ CHAR cIn, _Out_ LONG64 *pi64Out)                                                                                     |
| hti     | HRESULT       | Var18FromI2(_In_ SHORT sIn, _Out_ LONG64 *pi64Out)                                                                                    |
| fti     | HRESULT       | Var18FromR4(_In_ FLOAT fltIn, _Out_ LONG64 *pi64Out)                                                                                  |
| diti    | HRESULT       | Var18FromR8(_In_ DOUBLE dblIn, _Out_ LONG64 *pi64Out)                                                                                 |
| suiuiti | HRESULT       | Var18FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ unsigned long dwFlags, _Out_ LONG64 *pi64Out)                                 |
| ucti    | HRESULT       | Var18FromUI1(_In_ BYTE bIn, _Out_ LONG64 *pi64Out)                                                                                    |
| uhti    | HRESULT       | Var18FromUI2(_In_ USHORT uiIn, _Out_ LONG64 *pi64Out)                                                                                 |
| uiti    | HRESULT       | Var18FromUI4(_In_ ULONG ulIn, _Out_ LONG64 *pi64Out)                                                                                  |
| ui6ti   | HRESULT       | Var18FromUI8(_In_ ULONG64 ui64In, _Out_ LONG64 *pi64Out)                                                                              |
| titi    | VOID          | VARIANT_UserFree(_In_ unsigned long *pFlags, _In_ VARIANT *pVariant)                                                                  |
| tatt    | unsignedchar* | VARIANT_UserMarshal(_In_ unsigned long *pFlags, _Inout_ unsigned char *pBuffer, _In_ VARIANT *pVariant)                               |
| tuiti   | unsignedlong  | VARIANT_UserSize(_In_ unsigned long *pFlags, _In_ unsigned long Offset, _In_ VARIANT *pVariant)                                       |
| tatt    | unsignedchar* | VARIANT_UserUnmarshal(_In_ unsigned long *pFlags, _In_ unsigned char *pBuffer, _Out_ VARIANT *pVariant)                               |
| ttuuhu  | HRESULT       | VariantChangeType(_Out_ VARIANTARG *pvargDest, _In_ const VARIANTARG *pvarSrc, _In_ USHORT wFlags, _In_ VARTYPE vt)                   |
| ttuiuhu | HRESULT       | VariantChangeTypeEx(_Out_ VARIANTARG *pvargDest, _In_ const VARIANTARG *pvarSrc, _In_ LCID lcid, _In_ USHORT wFlags, _In_ VARTYPE vt) |
| ti      | HRESULT       | VariantClear(_Inout_ VARIANTARG *pvarg)                                                                                               |

|         |         |                                                                                                                           |
|---------|---------|---------------------------------------------------------------------------------------------------------------------------|
| tii     | HRESULT | VariantCopy(_Out_ VARIANTARG *pvargDest, _In_ const VARIANTARG *pvargSrc)                                                 |
| tii     | HRESULT | VariantCopyInd(_Out_ VARIANT *pvarDest, _In_ const VARIANTARG *pvargSrc)                                                  |
| ti      | VOID    | VariantInit(_Out_ VARIANTARG *pvarg)                                                                                      |
| dtii    | int     | VariantTimeToDosDateTime(_In_ DOUBLE vtime, _Out_ USHORT *pwDosDate, _Out_ USHORT *pwDosTime)                             |
| diti    | int     | VariantTimeToSystemTime(_In_ DOUBLE vtime, _Out_ LPSYSTEMTIME lpSystemTime)                                               |
| ttii    | HRESULT | VarDiv(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                     |
| ttii    | HRESULT | VarDiv(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                     |
| tii     | HRESULT | VarIn(_In_ LPVARIANT pvarIn, _Out_ LPVARIANT pvarResult)                                                                  |
| ttii    | HRESULT | VarMod(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                     |
| iiuiti  | HRESULT | VarMonthName(_In_ int iMonth, _In_ int fAbbrev, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut)                                 |
| ttii    | HRESULT | VarMul(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                     |
| tii     | HRESULT | VarNeg(_In_ LPVARIANT pvarIn, _Out_ LPVARIANT pvarResult)                                                                 |
| tii     | HRESULT | VarNot(_In_ LPVARIANT pvarIn, _Out_ LPVARIANT pvarResult)                                                                 |
| ttuiti  | HRESULT | VarNumFromParseNum(_In_ NUMPARSE *pnumprs, _In_ BYTE *rgbDig, _In_ ULONG dwVtBits, _Out_ VARIANT *pvar)                   |
| ttii    | HRESULT | VarO(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                       |
| suiuiti | HRESULT | VarParseNumFromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ NUMPARSE *pnumprs, _Out_ BYTE *rgbDig) |
| ttii    | HRESULT | VarPow(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                     |
| fdi     | HRESULT | VarR4CompRE(_In_ float fltLeft, _In_ double dblRight)                                                                     |
| hti     | HRESULT | VarR4FromBool(_In_ VARIANT_BOOL boolIn, _Out_ FLOAT *pfltOut)                                                             |
| i6ti    | HRESULT | VarR4FromC(_In_ CY cyIn, _Out_ FLOAT *pfltOut)                                                                            |
| diti    | HRESULT | VarR4FromDate(_In_ DATE dateIn, _Out_ FLOAT *pfltOut)                                                                     |
| tii     | HRESULT | VarR4FromDec(_In_ const DECIMAL *pdecIn, _Out_ FLOAT *pfltOut)                                                            |
|         |         | VarR4FromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_                                                                   |

|         |         |                                                                                               |
|---------|---------|-----------------------------------------------------------------------------------------------|
| tuiti   | HRESULT | Float *pfltOut)                                                                               |
| cti     | HRESULT | VarR4FromI1(_In_ CHAR cIn, _Out_ FLOAT *pfltOut)                                              |
| hti     | HRESULT | VarR4FromI2(_In_ SHORT sIn, _Out_ FLOAT *pfltOut)                                             |
| uiti    | HRESULT | VarR4FromI4(_In_ LONG lIn, _Out_ FLOAT *pfltOut)                                              |
| i6ti    | HRESULT | VarR4FromI8(_In_ LONG64 i64In, _Out_ FLOAT *pfltOut)                                          |
| dti     | HRESULT | VarR4FromR2(_In_ DOUBLE dblIn, _Out_ FLOAT *pfltOut)                                          |
| suiuiti | HRESULT | VarR4FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ FLOAT *pfltOut)  |
| ucti    | HRESULT | VarR4FromUI1(_In_ BYTE bIn, _Out_ FLOAT *pfltOut)                                             |
| uhti    | HRESULT | VarR4FromUI2(_In_ USHORT uiIn, _Out_ FLOAT *pfltOut)                                          |
| uiti    | HRESULT | VarR4FromUI4(_In_ ULONG ulIn, _Out_ FLOAT *pfltOut)                                           |
| ui6ti   | HRESULT | VarR4FromUI8(_In_ ULONG64 ui64In, _Out_ FLOAT *pfltOut)                                       |
| hti     | HRESULT | VarR8FromBool(_In_ VARIANT_BOOL boolIn, _Out_ DOUBLE *pdblOut)                                |
| i6ti    | HRESULT | VarR8FromC(_In_ CY cyIn, DOUBLE *pdblOut)                                                     |
| dti     | HRESULT | VarR8FromDate(_In_ DATE dateIn, _Out_ DOUBLE *pdblOut)                                        |
| tti     | HRESULT | VarR8FromDec(_In_ const DECIMAL *pdecIn, _Out_ DOUBLE *pdblOut)                               |
| tuiti   | HRESULT | VarR8FromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ DOUBLE *pdblOut)                      |
| cti     | HRESULT | VarR8FromI1(_In_ CHAR cIn, DOUBLE *pdblOut)                                                   |
| hti     | HRESULT | VarR8FromI2(_In_ SHORT sIn, _Out_ DOUBLE *pdblOut)                                            |
| uiti    | HRESULT | VarR8FromI4(_In_ LONG lIn, _Out_ DOUBLE *pdblOut)                                             |
| i6ti    | HRESULT | VarR8FromI8(_In_ LONG64 i64In, _Out_ DOUBLE *pdblOut)                                         |
| fti     | HRESULT | VarR8FromR4(_In_ FLOAT fltIn, _Out_ DOUBLE *pdblOut)                                          |
| suiuiti | HRESULT | VarR8FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ DOUBLE *pdblOut) |
| ucti    | HRESULT | VarR8FromUI1(_In_ BYTE bIn, _Out_ DOUBLE *pdblOut)                                            |
| uhti    | HRESULT | VarR8FromUI2(_In_ USHORT uiIn, _Out_ DOUBLE *pdblOut)                                         |
| uiti    | HRESULT | VarR8FromUI4(_In_ ULONG ulIn, _Out_ DOUBLE *pdblOut)                                          |
| ui6ti   | HRESULT | VarR8FromUI8(_In_ ULONG64 ui64In, _Out_ DOUBLE *pdblOut)                                      |
| ddti    | HRESULT | VarR8Pow(_In_ double dblLeft, _In_ double dblRight, _Out_ double *pdblResult)                 |
| diti    | HRESULT | VarR8Round(_In_ double dblIn, _In_ int cDecimals, _Out_ double *pdblResult)                   |
| titi    | HRESULT | VarRound(_In_ LPVARIANT pvarIn, _In_ int cDecimals, _Out_                                     |

|          |         |                                                                                                                                                                                |
|----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |         | LPVARIANT pvarResult)                                                                                                                                                          |
| ttti     | HRESULT | VarSub(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                                                                                          |
| stiiuiti | HRESULT | VarTokenizeFormatString(_In_opt_ LPOLESTR pstrFormat, _Inout_ LPBYTE rgbTok, _In_ int cbTok, _In_ int iFirstDay, _In_ int iFirstWeek, _In_ LCID lcid, _In_opt_ int *pcbActual) |
| duiti    | HRESULT | VarUdateFromDate(_In_ DATE dateIn, _In_ ULONG dwFlags, _Out_ UPDATE *pupdateOut)                                                                                               |
| hti      | HRESULT | VarUIFromBool(_In_ VARIANT_BOOL boolIn, _Out_ BYTE *pbOut)                                                                                                                     |
| i6ti     | HRESULT | VarUIFromCy(_In_ CY cyIn, _Out_ BYTE *pbOut)                                                                                                                                   |
| diti     | HRESULT | VarUIFromDate(_In_ DATE dateIn, _Out_ BYTE *pbOut)                                                                                                                             |
| tti      | HRESULT | VarUIFromDec(_In_ const DECIMAL *pdecIn, _Out_ BYTE *pbOut)                                                                                                                    |
| tuiti    | HRESULT | VarUIFromDisp(IDispatch *pdispIn, _In_ LCID lcid, _Out_ BYTE *pbOut)                                                                                                           |
| cti      | HRESULT | VarUIFromI1(_In_ CHAR cIn, _Out_ BYTE *pbOut)                                                                                                                                  |
| hti      | HRESULT | VarUIFromI2(_In_ SHORT sIn, _Out_ BYTE *pbOut)                                                                                                                                 |
| uiti     | HRESULT | VarUIFromI4(_In_ LONG lIn, _Out_ BYTE *pbOut)                                                                                                                                  |
| i6ti     | HRESULT | VarUIFromI8(_In_ LONG64 i64In, _Out_ BYTE *pbOut)                                                                                                                              |
| fti      | HRESULT | VarUIFromR4(_In_ FLOAT fltIn, _Out_ BYTE *pbOut)                                                                                                                               |
| diti     | HRESULT | VarUIFromR8(_In_ DOUBLE dblIn, _Out_ BYTE *pbOut)                                                                                                                              |
| suiuiti  | HRESULT | VarUIFromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ BYTE *pbOut)                                                                                      |
| uhti     | HRESULT | VarUIFromUI2(_In_ USHORT uiIn, _Out_ BYTE *pbOut)                                                                                                                              |
| uiti     | HRESULT | VarUIFromUI4(_In_ ULONG ulIn, _Out_ BYTE *pbOut)                                                                                                                               |
| ui6ti    | HRESULT | VarUIFromUI8(_In_ ULONG64 ui64In, _Out_ BYTE *pbOut)                                                                                                                           |
| hti      | HRESULT | VarUI2FromBool(_In_ VARIANT_BOOL boolIn, _Out_ USHORT *puiOut)                                                                                                                 |
| i6ti     | HRESULT | VarUI2FromCy(_In_ CY cyIn, _Out_ USHORT *puiOut)                                                                                                                               |
| diti     | HRESULT | VarUI2FromDate(_In_ DATE dateIn, _Out_ USHORT *puiOut)                                                                                                                         |
| tti      | HRESULT | VarUI2FromDec(_In_ const DECIMAL *pdecIn, _Out_ USHORT *puiOut)                                                                                                                |
| tuiti    | HRESULT | VarUI2FromDisp(_In_ IDispatch *pdispIn, _In_ LCID lcid, _Out_ USHORT *puiOut)                                                                                                  |
| cti      | HRESULT | VarUI2FromI1(_In_ CHAR cIn, _Out_ USHORT *puiOut)                                                                                                                              |
| hti      | HRESULT | VarUI2FromI2(_In_ SHORT sIn, _Out_ USHORT *puiOut)                                                                                                                             |
| uiti     | HRESULT | VarUI2FromI4(_In_ LONG lIn, _Out_ USHORT *puiOut)                                                                                                                              |

|         |         |                                                                                               |
|---------|---------|-----------------------------------------------------------------------------------------------|
| i6ti    | HRESULT | VarUI2FromI8(_In_ LONG64 i64In, _Out_ USHORT *puiOut)                                         |
| fti     | HRESULT | VarUI2FromR4(_In_ FLOAT fltIn, _Out_ USHORT *puiOut)                                          |
| dti     | HRESULT | VarUI2FromR8(_In_ DOUBLE dblIn, USHORT *puiOut)                                               |
| suiuiti | HRESULT | VarUI2FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ USHORT *puiOut) |
| ucti    | HRESULT | VarUI2FromUI1(_In_ BYTE bIn, _Out_ USHORT *puiOut)                                            |
| uiti    | HRESULT | VarUI2FromUI4(_In_ ULONG ulIn, _Out_ USHORT *puiOut)                                          |
| ui6ti   | HRESULT | VarUI2FromUI8(_In_ ULONG64 i64In, _Out_ USHORT *puiOut)                                       |
| hti     | HRESULT | VarUI4FromBool(_In_ VARIANT_BOOL boolIn, _Out_ ULONG *pulOut)                                 |
| i6ti    | HRESULT | VarUI4FromCy(_In_ CY cyIn, _Out_ ULONG *pulOut)                                               |
| dti     | HRESULT | VarUI4FromDate(_In_ DATE dateIn, _Out_ ULONG *pulOut)                                         |
| titi    | HRESULT | VarUI4FromDec(_In_ const DECIMAL *pdecIn, _Out_ ULONG *pulOut)                                |
| tuiti   | HRESULT | VarUI4FromDisp(_In_ IDispatch *pdispIn, _In_ LCID lcid, _Out_ ULONG *pulOut)                  |
| cti     | HRESULT | VarUI4FromI1(_In_ CHAR cIn, _Out_ ULONG *pulOut)                                              |
| hti     | HRESULT | VarUI4FromI2(_In_ SHORT uiIn, _Out_ ULONG *pulOut)                                            |
| uiti    | HRESULT | VarUI4FromI4(_In_ LONG lIn, _Out_ ULONG *pulOut)                                              |
| i6ti    | HRESULT | VarUI4FromI8(_In_ LONG64 i64In, _Out_ ULONG *pulOut)                                          |
| fti     | HRESULT | VarUI4FromR4(_In_ FLOAT fltIn, _Out_ ULONG *pulOut)                                           |
| dti     | HRESULT | VarUI4FromR8(_In_ DOUBLE dblIn, _Out_ ULONG *pulOut)                                          |
| suiuiti | HRESULT | VarUI4FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ ULONG dwFlags, _Out_ ULONG *pulOut)  |
| ucti    | HRESULT | VarUI4FromUI1(_In_ BYTE bIn, _Out_ ULONG *pulOut)                                             |
| uhti    | HRESULT | VarUI4FromUI2(_In_ USHORT uiIn, _Out_ ULONG *pulOut)                                          |
| ui6ti   | HRESULT | VarUI4FromUI8(_In_ ULONG64 ui64In, _Out_ ULONG *pulOut)                                       |
| hti     | HRESULT | VarUI8FromBool(_In_ VARIANT_BOOL boolIn, _Out_ ULONG64 *pi64Out)                              |
| i6ti    | HRESULT | VarUI8FromCy(_In_ CY cyIn, _Out_ ULONG64 *pi64Out)                                            |
| dti     | HRESULT | VarUI8FromDate(_In_ DATE dateIn, _Out_ ULONG64 *pi64Out)                                      |
| titi    | HRESULT | VarUI8FromDec(_In_ const DECIMAL *pdecIn, _Out_ ULONG64 *pi64Out)                             |
| tuiti   | HRESULT | VarUI8FromDisp(_In_ IDispatch *pdispIn, _In_ LCID lcid, _Out_ ULONG64 *pi64Out)               |
| cti     | HRESULT | VarUI8FromI1(_In_ CHAR cIn, _Out_ ULONG64 *pi64Out)                                           |
| hti     | HRESULT | VarUI8FromI2(_In_ SHORT sIn, _Out_ ULONG64 *pi64Out)                                          |

|         |         |                                                                                                                   |
|---------|---------|-------------------------------------------------------------------------------------------------------------------|
| i6ti    | HRESULT | VarUI8FromI6(_In_ LONG64 ui64In, _Out_ ULONG64 *pi64Out)                                                          |
| fti     | HRESULT | VarUI8FromR4(_In_ FLOAT fltIn, _Out_ ULONG64 *pi64Out)                                                            |
| dti     | HRESULT | VarUI8FromR8(_In_ DOUBLE dblIn, _Out_ ULONG64 *pi64Out)                                                           |
| suiuiti | HRESULT | VarUI8FromStr(_In_ LPCOLESTR strIn, _In_ LCID lcid, _In_ unsigned long dwFlags, _Out_ ULONG64 *pi64Out)           |
| ucti    | HRESULT | VarUI8FromUI1(_In_ BYTE bIn, _Out_ ULONG64 *pi64Out)                                                              |
| uhti    | HRESULT | VarUI8FromUI2(_In_ USHORT uiIn, _Out_ ULONG64 *pi64Out)                                                           |
| uiti    | HRESULT | VarUI8FromUI4(_In_ ULONG ulIn, _Out_ ULONG64 *pi64Out)                                                            |
| iiuuiti | HRESULT | VarWeekdayName(_In_ int iWeekday, _In_ int fAbbrev, _In_ int iFirstDay, _In_ ULONG dwFlags, _Out_ BSTR *pbstrOut) |
| ttti    | HRESULT | VarVar(_In_ LPVARIANT pvarLeft, _In_ LPVARIANT pvarRight, _Out_ LPVARIANT pvarResult)                             |
| sti     | HRESULT | VectorFromBstr(_In_ BSTR bstr, _Out_ SAFEARRAY **ppsa)                                                            |

# Opengl32.dll

|           |           |                                                                                                                            |
|-----------|-----------|----------------------------------------------------------------------------------------------------------------------------|
| ifi       | VOID      | glAccum(GLenum op, GLfloat value)                                                                                          |
| ifi       | VOID      | glAlphaFunc(GLenum func, GLclampf ref)                                                                                     |
| itti      | GLboolean | glAreTexturesResident(GLsizei n, const GLuint *textures, GLboolean *residences)                                            |
| ii        | VOID      | glArrayElement(GLint index)                                                                                                |
| ii        | VOID      | glBegin(GLenum mode)                                                                                                       |
| iuii      | VOID      | glBindTexture(GLenum target, GLuint texture)                                                                               |
| iiffffffi | VOID      | glBitmap(GLsizei width, GLsizei height, GLfloat xorig, GLfloat yorig, GLfloat xmove, GLfloat ymove, const GLubyte *bitmap) |
| iii       | VOID      | glBlendFunc(GLenum sfactor, GLenum dfactor)                                                                                |
| uii       | VOID      | glCallList(GLuint list)                                                                                                    |
| iiti      | VOID      | glCallLists(GLsizei n, GLenum type, const GLvoid *lists)                                                                   |
| ci        | VOID      | glClear(GLbitfield mask)                                                                                                   |
| ffffi     | VOID      | glClearAccum(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)                                                      |
| ffffi     | VOID      | glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)                                                  |
| di        | VOID      | glClearDepth(GLclampd depth)                                                                                               |
| fi        | VOID      | glClearIndex(GLfloat c)                                                                                                    |
| ii        | VOID      | glClearStencil(GLint s)                                                                                                    |
| iti       | VOID      | glClipPlane(GLenum plane, const GLdouble *equation)                                                                        |
| ccci      | VOID      | glColor3b(GLbyte red, GLbyte green, GLbyte blue)                                                                           |
| ti        | VOID      | glColor3bv(const GLbyte *v)                                                                                                |
| dddi      | VOID      | glColor3d(GLdouble red, GLdouble green, GLdouble blue)                                                                     |
| ti        | VOID      | glColor3dv(const GLdouble *v)                                                                                              |
| fffi      | VOID      | glColor3f(GLfloat red, GLfloat green, GLfloat blue)                                                                        |
| ti        | VOID      | glColor3fv(const GLfloat *v)                                                                                               |
| iiii      | VOID      | glColor3i(GLint red, GLint green, GLint blue)                                                                              |
| ti        | VOID      | glColor3iv(const GLint *v)                                                                                                 |
| hhhi      | VOID      | glColor3s(GLshort red, GLshort green, GLshort blue)                                                                        |
| ti        | VOID      | glColor3sv(const GLshort *v)                                                                                               |
| ccci      | VOID      | glColor3ub(GLubyte red, GLubyte green, GLubyte blue)                                                                       |
| ti        | VOID      | glColor3ubv(const GLubyte *v)                                                                                              |

|           |      |                                                                                                                                    |
|-----------|------|------------------------------------------------------------------------------------------------------------------------------------|
| uiuiuii   | VOID | glColor3ui (GLuint red, GLuint green, GLuint blue)                                                                                 |
| ti        | VOID | glColor3uiv(const GLuint *v)                                                                                                       |
| hhhi      | VOID | glColor3ui (GLushort red, GLushort green, GLushort blue)                                                                           |
| ti        | VOID | glColor3usv(const GLushort *v)                                                                                                     |
| cccci     | VOID | glColor4b (GLbyte red, GLbyte green, GLbyte blue, GLbyte alpha)                                                                    |
| ti        | VOID | glColor4bv(const GLbyte *v)                                                                                                        |
| ddddi     | VOID | glColor4d (GLdouble red, GLdouble green, GLdouble blue, GLdouble alpha)                                                            |
| ti        | VOID | glColor4dv(const GLdouble *v)                                                                                                      |
| ffffi     | VOID | glColor4f (GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)                                                                |
| ti        | VOID | glColor4fv(const GLfloat *v)                                                                                                       |
| iiii      | VOID | glColor4i (GLint red, GLint green, GLint blue, GLint alpha)                                                                        |
| ti        | VOID | glColor4iv(const GLint *v)                                                                                                         |
| hhhi      | VOID | glColor4i (GLshort red, GLshort green, GLshort blue, GLshort alpha)                                                                |
| ti        | VOID | glColor4sv(const GLshort *v)                                                                                                       |
| cccci     | VOID | glColor4ub (GLubyte red, GLubyte green, GLubyte blue, GLubyte alpha)                                                               |
| ti        | VOID | glColor4ubv(const GLubyte *v)                                                                                                      |
| uiuiuiuii | VOID | glColor4ui (GLuint red, GLuint green, GLuint blue, GLuint alpha)                                                                   |
| ti        | VOID | glColor4uiv(const GLuint *v)                                                                                                       |
| hhhi      | VOID | glColor4ui (GLushort red, GLushort green, GLushort blue, GLushort alpha)                                                           |
| ti        | VOID | glColor4usv(const GLushort *v)                                                                                                     |
| iiii      | VOID | glColorMask(GLboolean red, GLboolean green, GLboolean blue, GLboolean alpha)                                                       |
| ii        | VOID | glColorMaterial(GLenum face, GLenum mode)                                                                                          |
| iiiti     | VOID | glColorPointer(GLint size, GLenum type, GLsizei stride, const GLvoid *pointer)                                                     |
| iiiiii    | VOID | glCopyPixels(GLint x, GLint y, GLsizei width, GLsizei height, GLenum type)                                                         |
| iiiiiiii  | VOID | glCopyTexImage1D(GLenum target, GLint level, GLenum internalFormat, GLint x, GLint y, GLsizei width, GLint border)                 |
| iiiiiiii  | VOID | glCopyTexImage2D(GLenum target, GLint level, GLenum internalFormat, GLint x, GLint y, GLsizei width, GLsizei height, GLint border) |
| iiiiii    | VOID | glCopyTexSubImage1D(GLenum target, GLint level, GLint xoffset, GLint x, GLint y, GLsizei width)                                    |
| iiiiiiii  | VOID | glCopyTexSubImage2D(GLenum target, GLint level, GLint xoffset, GLint yoffset, GLint x, GLint y, GLsizei width, GLsizei height)     |
| ii        | VOID | glCullFace(GLenum mode)                                                                                                            |
| uiii      | VOID | glDeleteLists(GLuint list, GLsizei range)                                                                                          |

|         |      |                                                                                               |
|---------|------|-----------------------------------------------------------------------------------------------|
| iti     | VOID | glDeleteTextures(GLsizei n, const GLuint *textures)                                           |
| ii      | VOID | glDepthFunc(GLenum func)                                                                      |
| ii      | VOID | glDepthMask(GLboolean flag)                                                                   |
| ddi     | VOID | glDepthRange(GLclampd zNear, GLclampd zFar)                                                   |
| ii      | VOID | glDisable(GLenum cap)                                                                         |
| ii      | VOID | glDisableClientState(GLenum array)                                                            |
| iiii    | VOID | glDrawArrays(GLenum mode, GLint first, GLsizei count)                                         |
| ii      | VOID | glDrawBuffer(GLenum mode)                                                                     |
| iiiti   | VOID | glDrawElements(GLenum mode, GLsizei count, GLenum type, const GLvoid *indices)                |
| iiiiiti | VOID | glDrawPixels(GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels) |
| ii      | VOID | glEdgeFlag(GLboolean flag)                                                                    |
| iti     | VOID | glEdgeFlagPointer(GLsizei stride, const GLvoid *pointer)                                      |
| ti      | VOID | glEdgeFlag(const GLboolean *flag)                                                             |
| ii      | VOID | glEnable(GLenum cap)                                                                          |
| ii      | VOID | glEnableClientState(GLenum array)                                                             |
| i       | VOID | glEnd(void)                                                                                   |
| i       | VOID | glEndList(void)                                                                               |
| di      | VOID | glEvalCoord1d(GLdouble u)                                                                     |
| ti      | VOID | glEvalCoord1dv(const GLdouble *u)                                                             |
| fi      | VOID | glEvalCoord1f(GLfloat u)                                                                      |
| ti      | VOID | glEvalCoord1fv(const GLfloat *u)                                                              |
| ddi     | VOID | glEvalCoord2d(GLdouble u, GLdouble v)                                                         |
| ti      | VOID | glEvalCoord2dv(const GLdouble *u)                                                             |
| ffi     | VOID | glEvalCoord2f(GLfloat u, GLfloat v)                                                           |
| ti      | VOID | glEvalCoord2fv(const GLfloat *u)                                                              |
| iiii    | VOID | glEvalMesh1(GLenum mode, GLint i1, GLint i2)                                                  |
| iiiiii  | VOID | glEvalMesh2(GLenum mode, GLint i1, GLint i2, GLint j1, GLint j2)                              |
| ii      | VOID | glEvalPoint1(GLint i)                                                                         |
| iii     | VOID | glEvalPoint2(GLint i, GLint j)                                                                |
| iiiti   | VOID | glFeedbackBuffer(GLsizei size, GLenum type, GLfloat *buffer)                                  |
| i       | VOID | glFinish(void)                                                                                |
| i       | VOID | glFlush(void)                                                                                 |
| ifi     | VOID | glFog(GLenum pname, GLfloat param)                                                            |

|         |          |                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------|
| iti     | VOID     | glFogfv(GLenum pname, const GLfloat *params)                                                           |
| iii     | VOID     | glFogf(GLenum pname, GLint param)                                                                      |
| iti     | VOID     | glFogiv(GLenum pname, const GLint *params)                                                             |
| ii      | VOID     | glFrontFace(GLenum mode)                                                                               |
| ddddddi | VOID     | glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar) |
| iui     | GLuint   | glGenLists(GLsizei range)                                                                              |
| iti     | VOID     | glGenTextures(GLsizei n, GLuint *textures)                                                             |
| iti     | VOID     | glGetBooleanv(GLenum pname, GLboolean *params)                                                         |
| iti     | VOID     | glGetClipPlane(GLenum plane, GLdouble *equation)                                                       |
| iti     | VOID     | glGetDoublev(GLenum pname, GLboolean *params)                                                          |
| i       | GLenum   | glGetError(void)                                                                                       |
| iti     | VOID     | glGetFloatv(GLenum pname, GLboolean *params)                                                           |
| iti     | VOID     | glGetIntegerv(GLenum pname, GLboolean *params)                                                         |
| iiti    | VOID     | glGetLightfv(GLenum light, GLenum pname, GLfloat *params)                                              |
| iiti    | VOID     | glGetLightiv(GLenum light, GLenum pname, GLint *params)                                                |
| iiti    | VOID     | glGetMapdv(GLenum target, GLenum query, GLdouble *v)                                                   |
| iiti    | VOID     | glGetMapfv(GLenum target, GLenum query, GLfloat *v)                                                    |
| iiti    | VOID     | glGetMapiv(GLenum target, GLenum query, GLint *v)                                                      |
| iiti    | VOID     | glGetMaterialfv(GLenum face, GLenum pname, GLfloat *params)                                            |
| iiti    | VOID     | glGetMaterialiv(GLenum face, GLenum pname, GLint *params)                                              |
| iti     | VOID     | glGetPixelMapfv(GLenum map, GLfloat *values)                                                           |
| iti     | VOID     | glGetPixelMapuiv(GLenum map, GLuint *values)                                                           |
| iti     | VOID     | glGetPixelMapusv(GLenum map, GLushort *values)                                                         |
| iti     | VOID     | glGetPointerv(GLenum pname, GLvoid **params)                                                           |
| ti      | VOID     | glGetPolygonSpple(GLubyte *mask)                                                                       |
| it      | GLubyte* | glGetString(GLenum name)                                                                               |
| iiti    | VOID     | glGetTexEnvfv(GLenum target, GLenum pname, GLfloat *params)                                            |
| iiti    | VOID     | glGetTexEnviv(GLenum target, GLenum pname, GLint *params)                                              |
| iiti    | VOID     | glGetTexGendv(GLenum coord, GLenum pname, GLdouble *params)                                            |
| iiti    | VOID     | glGetTexGenfv(GLenum coord, GLenum pname, GLfloat *params)                                             |
| iiti    | VOID     | glGetTexGeniv(GLenum coord, GLenum pname, GLint *params)                                               |
| iiiti   | VOID     | glGetTexImage(GLenum target, GLint level, GLenum format, GLenum type, GLvoid *pixels)                  |
| iiiti   | VOID     | glGetTexLevelParameterfv(GLenum target, GLint level, GLenum pname, GLfloat *params)                    |

|       |           |                                                                                   |
|-------|-----------|-----------------------------------------------------------------------------------|
|       |           |                                                                                   |
| iiiti | VOID      | glGetTexLevelParameteriv(GLenum target, GLint level, GLenum pname, GLint *params) |
| iiti  | VOID      | glGetTexParameterfv(GLenum target, GLenum pname, GLfloat *params)                 |
| iiti  | VOID      | glGetTexParameteriv(GLenum target, GLenum pname, GLint *params)                   |
| iii   | VOID      | glHint(GLenum target, GLenum mode)                                                |
| di    | VOID      | glIndexd(GLdouble c)                                                              |
| ti    | VOID      | glIndexdv(const GLdouble *c)                                                      |
| fi    | VOID      | glIndexf(GLfloat c)                                                               |
| ti    | VOID      | glIndexfv(const GLfloat *c)                                                       |
| ii    | VOID      | glIndexi(GLint c)                                                                 |
| ti    | VOID      | glIndexiv(const GLint *c)                                                         |
| uii   | VOID      | glIndexMask(GLuint mask)                                                          |
| iiti  | VOID      | glIndexPointer(GLenum type, GLsizei stride, const GLvoid *pointer)                |
| hi    | VOID      | glIndexs(GLshort c)                                                               |
| ti    | VOID      | glIndexsv(const GLshort *c)                                                       |
| i     | VOID      | glInvalidateNames(void)                                                           |
| iiiti | VOID      | glInterleavedArrays(GLenum format, GLsizei stride, const GLvoid *pointer)         |
| ii    | GLboolean | glIsEnabled(GLenum cap)                                                           |
| uii   | GLboolean | glIsList(GLuint list)                                                             |
| uii   | GLboolean | glIsTexture(GLuint texture)                                                       |
| iiifi | VOID      | glLight(GLenum light, GLenum pname, GLfloat param)                                |
| iiti  | VOID      | glLightfv(GLenum light, GLenum pname, const GLfloat *params)                      |
| iiiii | VOID      | glLight(GLenum light, GLenum pname, GLint param)                                  |
| iiti  | VOID      | glLightiv(GLenum light, GLenum pname, const GLint *params)                        |
| iti   | VOID      | glLightModelf(GLenum pname, GLfloat *param)                                       |
| iti   | VOID      | glLightModelfv(GLenum pname, const GLfloat *params)                               |
| iii   | VOID      | glLightModeli(GLenum pname, GLint param)                                          |
| iti   | VOID      | glLightModeliv(GLenum pname, const GLint *params)                                 |
| ihii  | VOID      | glLineStipple(GLint factor, GLushort pattern)                                     |
| fi    | VOID      | glLineWidth(GLfloat width)                                                        |
| uii   | VOID      | glListBase(GLuint base)                                                           |
| i     | VOID      | glLoadIdentity(void)                                                              |
| ti    | VOID      | glLoadMatrixd(const GLdouble *m)                                                  |
| ti    | VOID      | glLoadMatrixf(const GLfloat *m)                                                   |

|             |      |                                                                                                                                                              |
|-------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uui         | VOID | glLoadName(GLuint name)                                                                                                                                      |
| ii          | VOID | glLogicOp(GLenum opcode)                                                                                                                                     |
| iddiiti     | VOID | glMap1d(GLenum target, GLdouble u1, GLdouble u2, GLint stride, GLint order, const GLdouble *points)                                                          |
| iffiiti     | VOID | glMap1f(GLenum target, GLfloat u1, GLfloat u2, GLint stride, GLint order, const GLfloat *points)                                                             |
| iddiiddiiti | VOID | glMap2d(GLenum target, GLdouble u1, GLdouble u2, GLint ustride, GLint uorder, GLdouble v1, GLdouble v2, GLint vstride, GLint vorder, const GLdouble *points) |
| iffiiffiiti | VOID | glMap2f(GLenum target, GLfloat u1, GLfloat u2, GLint ustride, GLint uorder, GLfloat v1, GLfloat v2, GLint vstride, GLint vorder, const GLfloat *points)      |
| iddi        | VOID | glMapGrid1d(GLint un, GLdouble u1, GLdouble u2)                                                                                                              |
| iffi        | VOID | glMapGrid1f(GLint un, GLfloat u1, GLfloat u2)                                                                                                                |
| iddiddi     | VOID | glMapGrid2d(GLint un, GLdouble u1, GLdouble u2, GLint vn, GLdouble v1, GLdouble v2)                                                                          |
| iffiffi     | VOID | glMapGrid2f(GLint un, GLfloat u1, GLfloat u2, GLint vn, GLfloat v1, GLfloat v2)                                                                              |
| iifi        | VOID | glMaterialf(GLenum face, GLenum pname, GLfloat param)                                                                                                        |
| iiti        | VOID | glMaterialfv(GLenum face, GLenum pname, const GLfloat *params)                                                                                               |
| iiii        | VOID | glMateriali(GLenum face, GLenum pname, GLint param)                                                                                                          |
| iiti        | VOID | glMaterialiv(GLenum face, GLenum pname, const GLint *params)                                                                                                 |
| ii          | VOID | glMatrixMode(GLenum mode)                                                                                                                                    |
| ti          | VOID | glMultiMatrixd(const GLdouble *m)                                                                                                                            |
| ti          | VOID | glMultiMatrixf(const GLdouble *m)                                                                                                                            |
| uiii        | VOID | glNewList(GLuint list, GLenum mode)                                                                                                                          |
| ccci        | VOID | glNormal3b(GLbyte nx, GLbyte ny, GLbyte nz)                                                                                                                  |
| ti          | VOID | glNormal3bv(const GLbyte *v)                                                                                                                                 |
| dddi        | VOID | glNormal3d(GLdouble nx, GLdouble ny, GLdouble nz)                                                                                                            |
| ti          | VOID | glNormal3dv(const GLdouble *v)                                                                                                                               |
| fffi        | VOID | glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz)                                                                                                               |
| ti          | VOID | glNormal3fv(const GLfloat *v)                                                                                                                                |
| iiii        | VOID | glNormal3i(GLint nx, GLint ny, GLint nz)                                                                                                                     |
| ti          | VOID | glNormal3iv(const GLint *v)                                                                                                                                  |
| hhhi        | VOID | glNormal3s(GLshort nx, GLshort ny, GLshort nz)                                                                                                               |
| ti          | VOID | glNormal3sv(const GLshort *v)                                                                                                                                |
| iiti        | VOID | glNormalPointer(GLenum type, GLsizei stride, const GLvoid *pointer)                                                                                          |

|         |      |                                                                                                      |
|---------|------|------------------------------------------------------------------------------------------------------|
| ddddddi | VOID | glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar) |
| fi      | VOID | glPassThrough(GLfloat token)                                                                         |
| iiti    | VOID | glPixelMapfv(GLenum map, GLsizei mapsize, const GLfloat *values)                                     |
| iiti    | VOID | glPixelMapuiv(GLenum map, GLsizei mapsize, const GLuint *values)                                     |
| iiti    | VOID | glPixelMapsv(GLenum map, GLsizei mapsize, const GLushort *values)                                    |
| ifi     | VOID | glPixelStoref(GLenum pname, GLfloat param)                                                           |
| iii     | VOID | glPixelStorei(GLenum pname, GLint param)                                                             |
| ifi     | VOID | glPixelTransferf(GLenum pname, GLfloat param)                                                        |
| iii     | VOID | glPixelTransferi(GLenum pname, GLint param)                                                          |
| ffi     | VOID | glPixelZoom(GLfloat xfactor, GLfloat yfactor)                                                        |
| fi      | VOID | glPointSize(GLfloat size)                                                                            |
| iii     | VOID | glPolygonMode(GLenum face, GLenum mode)                                                              |
| ffi     | VOID | glPolygonOffset(GLfloat factor, GLfloat units)                                                       |
| ti      | VOID | glPolygonStipple(const GLubyte *mask)                                                                |
| i       | VOID | glPopAttrib(void)                                                                                    |
| i       | VOID | glPopClientAttrib(void)                                                                              |
| i       | VOID | glPopMatrix(void)                                                                                    |
| i       | VOID | glPopName(void)                                                                                      |
| itti    | VOID | glPrioritizeTextures(GLsizei n, const GLuint *textures, const GLclampf *priorities)                  |
| ci      | VOID | glPushAttrib(GLbitfield mask)                                                                        |
| ci      | VOID | glPushClientAttrib(GLbitfield mask)                                                                  |
| i       | VOID | glPushMatrix(void)                                                                                   |
| uui     | VOID | glPushName(GLuint name)                                                                              |
| ddi     | VOID | glRasterPos2d(GLdouble x, GLdouble y)                                                                |
| ti      | VOID | glRasterPos2dv(const GLdouble *v)                                                                    |
| ffi     | VOID | glRasterPos2f(GLfloat x, GLfloat y)                                                                  |
| ti      | VOID | glRasterPos2fv(const GLfloat *v)                                                                     |
| iii     | VOID | glRasterPos2i(GLint x, GLint y)                                                                      |
| ti      | VOID | glRasterPos2iv(const GLint *v)                                                                       |
| hhi     | VOID | glRasterPos2s(GLshort x, GLshort y)                                                                  |
| ti      | VOID | glRasterPos2sv(const GLshort *v)                                                                     |
| dddi    | VOID | glRasterPos3d(GLdouble x, GLdouble y, GLdouble z)                                                    |
| ti      | VOID | glRasterPos3dv(const GLdouble *v)                                                                    |

|         |       |                                                                                                           |
|---------|-------|-----------------------------------------------------------------------------------------------------------|
| ffff    | VOID  | glRasterPos3f(GLfloat x, GLfloat y, GLfloat z)                                                            |
| ti      | VOID  | glRasterPos3fv(const GLfloat *v)                                                                          |
| iiii    | VOID  | glRasterPos3i(GLint x, GLint y, GLint z)                                                                  |
| ti      | VOID  | glRasterPos3iv(const GLint *v)                                                                            |
| hhhi    | VOID  | glRasterPos3s(GLshort x, GLshort y, GLshort z)                                                            |
| ti      | VOID  | glRasterPos3sv(const GLshort *v)                                                                          |
| ddddi   | VOID  | glRasterPos4d(GLdouble x, GLdouble y, GLdouble z, GLdouble w)                                             |
| ti      | VOID  | glRasterPos4dv(const GLdouble *v)                                                                         |
| ffffi   | VOID  | glRasterPos4f(GLfloat x, GLfloat y, GLfloat z, GLfloat w)                                                 |
| ti      | VOID  | glRasterPos4fv(const GLfloat *v)                                                                          |
| iiiiii  | VOID  | glRasterPos4i(GLint x, GLint y, GLint z, GLint w)                                                         |
| ti      | VOID  | glRasterPos4iv(const GLint *v)                                                                            |
| hhghi   | VOID  | glRasterPos4s(GLshort x, GLshort y, GLshort z, GLshort w)                                                 |
| ti      | VOID  | glRasterPos4sv(const GLshort *v)                                                                          |
| ii      | VOID  | glReadBuffer(GLenum mode)                                                                                 |
| iiiiiti | VOID  | glReadPixels(GLint x, GLint y, GLsizei width, GLsizei height, GLenum format, GLenum type, GLvoid *pixels) |
| ddddi   | VOID  | glRectd(GLdouble x1, GLdouble y1, GLdouble x2, GLdouble y2)                                               |
| tii     | VOID  | glRectdv(const GLdouble *v1, const GLdouble *v2)                                                          |
| ffffi   | VOID  | glRectf(GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2)                                                   |
| tii     | VOID  | glRectfv(const GLfloat *v1, const GLfloat *v2)                                                            |
| iiiiii  | VOID  | glRecti(GLint x1, GLint y1, GLint x2, GLint y2)                                                           |
| tii     | VOID  | glRectiv(const GLint *v1, const GLint *v2)                                                                |
| hhghi   | VOID  | glRects(GLshort x1, GLshort y1, GLshort x2, GLshort y2)                                                   |
| tii     | VOID  | glRectsv(const GLshort *v1, const GLshort *v2)                                                            |
| ii      | GLint | glRenderMode(GLenum mode)                                                                                 |
| ddddi   | VOID  | glRotated(GLdouble angle, GLdouble x, GLdouble y, GLdouble z)                                             |
| ffffi   | VOID  | glRotatedf(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)                                                |
| dddi    | VOID  | glScaled(GLdouble x, GLdouble y, GLdouble z)                                                              |
| ffffi   | VOID  | glScaledf(GLfloat x, GLfloat y, GLfloat z)                                                                |
| iiiiii  | VOID  | glScissor(GLint x, GLint y, GLsizei width, GLsizei height)                                                |
| iti     | VOID  | glSelectBuffer(GLsizei size, GLuint *buffer)                                                              |
| ii      | VOID  | glShadeModel(GLenum mode)                                                                                 |
| iiiiii  | VOID  | glStencilFunc(GLenum func, GLint ref, GLuint mask)                                                        |

|        |      |                                                                         |
|--------|------|-------------------------------------------------------------------------|
| uui    | VOID | glStencilMask(GLuint mask)                                              |
| iiii   | VOID | glStencilOp(GLenum fail, GLenum zfail, GLenum zpass)                    |
| di     | VOID | glTexCoord1d(GLdouble s)                                                |
| ti     | VOID | glTexCoord1dv(const GLdouble *v)                                        |
| fi     | VOID | glTexCoord1f(GLfloat s)                                                 |
| ti     | VOID | glTexCoord1fv(const GLfloat *v)                                         |
| ii     | VOID | glTexCoord1i(GLint s)                                                   |
| ti     | VOID | glTexCoord1iv(const GLint *v)                                           |
| hi     | VOID | glTexCoord1s(GLshort s)                                                 |
| ti     | VOID | glTexCoord1sv(const GLshort *v)                                         |
| ddi    | VOID | glTexCoord2d(GLdouble s, GLdouble t)                                    |
| ti     | VOID | glTexCoord2dv(const GLdouble *v)                                        |
| ti     | VOID | glTexCoord2fv(const GLfloat *v)                                         |
| iii    | VOID | glTexCoord2i(GLint s, GLint t)                                          |
| ti     | VOID | glTexCoord2iv(const GLint *v)                                           |
| hhi    | VOID | glTexCoord2s(GLshort s, GLshort t)                                      |
| ti     | VOID | glTexCoord2sv(const GLshort *v)                                         |
| dddi   | VOID | glTexCoord3d(GLdouble s, GLdouble t, GLdouble r)                        |
| ti     | VOID | glTexCoord3dv(const GLdouble *v)                                        |
| fffi   | VOID | glTexCoord3f(GLfloat s, GLfloat t, GLfloat r)                           |
| ti     | VOID | glTexCoord3fv(const GLfloat *v)                                         |
| iiii   | VOID | glTexCoord3i(GLint s, GLint t, GLint r)                                 |
| ti     | VOID | glTexCoord3iv(const GLint *v)                                           |
| hhhi   | VOID | glTexCoord3s(GLshort s, GLshort t, GLshort r)                           |
| ti     | VOID | glTexCoord3sv(const GLshort *v)                                         |
| ddddi  | VOID | glTexCoord4d(GLdouble s, GLdouble t, GLdouble r, GLdouble q)            |
| ti     | VOID | glTexCoord4dv(const GLdouble *v)                                        |
| ffffi  | VOID | glTexCoord4f(GLfloat s, GLfloat t, GLfloat r, GLfloat q)                |
| ti     | VOID | glTexCoord4fv(const GLfloat *v)                                         |
| iiiiii | VOID | glTexCoord4i(GLint s, GLint t, GLint r, GLint q)                        |
| ti     | VOID | glTexCoord4iv(const GLint *v)                                           |
| hhhhi  | VOID | glTexCoord4s(GLshort s, GLshort t, GLshort r, GLshort q)                |
| ti     | VOID | glTexCoord4sv(const GLshort *v)                                         |
| iiiti  | VOID | glTexCoordPointer(GLint size, GLenum type, GLsizei stride, const GLvoid |

|         |      |                                                                                                                                                              |
|---------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |      | *pointer)                                                                                                                                                    |
| iifi    | VOID | glTexEnvf(GLenum target, GLenum pname, GLfloat param)                                                                                                        |
| iiti    | VOID | glTexEnvfv(GLenum target, GLenum pname, const GLfloat *params)                                                                                               |
| iiii    | VOID | glTexEnvf(GLenum target, GLenum pname, GLint param)                                                                                                          |
| iiti    | VOID | glTexEnviv(GLenum target, GLenum pname, const GLint *params)                                                                                                 |
| iidi    | VOID | glTexGend(GLenum coord, GLenum pname, GLdouble param)                                                                                                        |
| iiti    | VOID | glTexGendv(GLenum coord, GLenum pname, const GLdouble *params)                                                                                               |
| iifi    | VOID | glTexGenf(GLenum coord, GLenum pname, GLfloat param)                                                                                                         |
| iiti    | VOID | glTexGenfv(GLenum coord, GLenum pname, const GLfloat *params)                                                                                                |
| iiii    | VOID | glTexGenf(GLenum coord, GLenum pname, GLint param)                                                                                                           |
| iiti    | VOID | glTexGeniv(GLenum coord, GLenum pname, const GLint *params)                                                                                                  |
| iiiiiti | VOID | glTexImage1D(GLenum target, GLint level, GLint internalformat, GLsizei width, GLint border, GLint format, GLenum type, const GLvoid *pixels)                 |
| iiiiiti | VOID | glTexImage2D(GLenum target, GLint level, GLint internalformat, GLsizei width, GLsizei height, GLint border, GLint format, GLenum type, const GLvoid *pixels) |
| iifi    | VOID | glTexParameterf(GLenum target, GLenum pname, GLfloat param)                                                                                                  |
| iiti    | VOID | glTexParameterfv(GLenum target, GLenum pname, const GLfloat *params)                                                                                         |
| iiii    | VOID | glTexParameterf(GLenum target, GLenum pname, GLint param)                                                                                                    |
| iiti    | VOID | glTexParameteriv(GLenum target, GLenum pname, const GLint *params)                                                                                           |
| iiiiiti | VOID | glTexSubImage1D(GLenum target, GLint level, GLint xoffset, GLsizei width, GLenum format, GLenum type, const GLvoid *pixels)                                  |
| iiiiiti | VOID | glTexSubImage2D(GLenum target, GLint level, GLint xoffset, GLint yoffset, GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels)   |
| dddi    | VOID | glTranslated(GLdouble x, GLdouble y, GLdouble z)                                                                                                             |
| ffi     | VOID | glTranslated(GLfloat x, GLfloat y, GLfloat z)                                                                                                                |
| ddi     | VOID | glVertex2d(GLdouble x, GLdouble y)                                                                                                                           |
| ti      | VOID | glVertex2dv(const GLdouble *v)                                                                                                                               |
| ffi     | VOID | glVertex2f(GLfloat x, GLfloat y)                                                                                                                             |
| ti      | VOID | glVertex2fv(const GLfloat *v)                                                                                                                                |
| iii     | VOID | glVertex2i(GLint x, GLint y)                                                                                                                                 |
| ti      | VOID | glVertex2iv(const GLint *v)                                                                                                                                  |
| hhi     | VOID | glVertex2s(GLshort x, GLshort y)                                                                                                                             |
| ti      | VOID | glVertex2sv(const GLshort *v)                                                                                                                                |
| dddi    | VOID | glVertex3d(GLdouble x, GLdouble y, GLdouble z)                                                                                                               |

|         |       |                                                                                                             |
|---------|-------|-------------------------------------------------------------------------------------------------------------|
| ti      | VOID  | glVertex3dv(const GLdouble *v)                                                                              |
| fffi    | VOID  | glVertex3f(GLfloat x, GLfloat y, GLfloat z)                                                                 |
| ti      | VOID  | glVertex3fv(const GLfloat *v)                                                                               |
| iiii    | VOID  | glVertex3i(GLint x, GLint y, GLint z)                                                                       |
| ti      | VOID  | glVertex3iv(const GLint *v)                                                                                 |
| hhhi    | VOID  | glVertex3s(GLshort x, GLshort y, GLshort z)                                                                 |
| ti      | VOID  | glVertex3sv(const GLshort *v)                                                                               |
| ddddi   | VOID  | glVertex4d(GLdouble x, GLdouble y, GLdouble z, GLdouble w)                                                  |
| ti      | VOID  | glVertex4dv(const GLdouble *v)                                                                              |
| ffffi   | VOID  | glVertex4f(GLfloat x, GLfloat y, GLfloat z, GLfloat w)                                                      |
| ti      | VOID  | glVertex4fv(const GLfloat *v)                                                                               |
| iiii    | VOID  | glVertex4i(GLint x, GLint y, GLint z, GLint w)                                                              |
| ti      | VOID  | glVertex4iv(const GLint *v)                                                                                 |
| hhhi    | VOID  | glVertex4s(GLshort x, GLshort y, GLshort z, GLshort w)                                                      |
| ti      | VOID  | glVertex4sv(const GLshort *v)                                                                               |
| iiiti   | VOID  | glVertexPointer(GLint size, GLenum type, GLsizei stride, const GLvoid *pointer)                             |
| iiii    | VOID  | glViewport(GLint x, GLint y, GLsizei width, GLsizei height)                                                 |
| ttuii   | BOOL  | wglCopyContext(HGLRC hglrcSrc, HGLRC hglrcDst, UINT mask)                                                   |
| tt      | HGLRC | wglCreateContext(HDC hdc)                                                                                   |
| tit     | HGLRC | wglCreateLayerContext(HDC hdc, int iLayerPlane)                                                             |
| ti      | BOOL  | wglDeleteContext(HGLRC hglrc)                                                                               |
| tiuiiti | BOOL  | wglDescribeLayerPlane(HDC hdc, int iPixelFormat, int iLayerPlane, UINT nBytes, LPLAYERPLANEDESCRIPTOR plpd) |
| t       | HGLRC | wglGetCurrentContext(void)                                                                                  |
| t       | HDC   | wglGetCurrentDC(void)                                                                                       |
| tiiti   | int   | wglGetLayerPaletteEntries(HDC hdc, int iLayerPlane, int iStart, int cEntries, COLORREF *pcr)                |
| at      | PROC  | wglGetProcAddress(LPCSTR lpszProc)                                                                          |
| tii     | BOOL  | wglMakeCurrent(HDC hdc, HGLRC hglrc)                                                                        |
| tiii    | BOOL  | wglRealizeLayerPalette(HDC hdc, int iLayerPlane, BOOL bRealize)                                             |
| tiiti   | int   | wglSetLayerPaletteEntries(HDC hdc, int iLayerPlane, int iStart, int cEntries, const COLORREF *pcr)          |
| tti     | BOOL  | wglShareList(HGLRC hglrc1, HGLRC hglrc2)                                                                    |
| tuii    | BOOL  | wglSwapLayerBuffers(HDC hdc, UINT fuPlanes)                                                                 |

|             |      |                                                                                                                                                                  |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuiiii    | BOOL | <a href="#">wglUseFontBitmaps</a> (HDC hdc, DWORD first, DWORD count, DWORD listBase)                                                                            |
| tuiuiiii    | BOOL | <a href="#">wglUseFontBitmapsA</a> (HDC hdc, DWORD first, DWORD count, DWORD listBase)                                                                           |
| tuiuiiii    | BOOL | <a href="#">wglUseFontBitmapsW</a> (HDC hdc, DWORD first, DWORD count, DWORD listBase)                                                                           |
| tuiuiiffiti | BOOL | <a href="#">wglUseFontOutlines</a> (HDC hdc, DWORD first, DWORD count, DWORD listBase, FLOAT deviation, FLOAT extrusion, int format, LPGLYPHMETRICSFLOAT lpgmf)  |
| tuiuiiffiti | BOOL | <a href="#">wglUseFontOutlinesA</a> (HDC hdc, DWORD first, DWORD count, DWORD listBase, FLOAT deviation, FLOAT extrusion, int format, LPGLYPHMETRICSFLOAT lpgmf) |
| tuiuiiffiti | BOOL | <a href="#">wglUseFontOutlinesW</a> (HDC hdc, DWORD first, DWORD count, DWORD listBase, FLOAT deviation, FLOAT extrusion, int format, LPGLYPHMETRICSFLOAT lpgmf) |

## Rasapi32.dll

|            |       |                                                                                                                                                                                                           |
|------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tui        | DWORD | RasClearConnectionStatistics(_In_ HRASCONN hRasConn)                                                                                                                                                      |
| tuiui      | DWORD | RasClearLinkStatistics(_In_ HRASCONN hRasConn, _In_ DWORD dwSubEntry)                                                                                                                                     |
| ttuiui     | DWORD | RasConnectionNotification(_In_ HRASCONN hrasconn, _In_ HANDLE hEvent, _In_ DWORD dwFlags)                                                                                                                 |
| ttuiui     | DWORD | RasConnectionNotificationA(_In_ HRASCONN hrasconn, _In_ HANDLE hEvent, _In_ DWORD dwFlags)                                                                                                                |
| ttuiui     | DWORD | RasConnectionNotificationW(_In_ HRASCONN hrasconn, _In_ HANDLE hEvent, _In_ DWORD dwFlags)                                                                                                                |
| tsui       | DWORD | RasCreatePhonebookEntry(_In_ HWND hwnd, _In_ LPCTSTR lpszPhonebook)                                                                                                                                       |
| taui       | DWORD | RasCreatePhonebookEntryA(_In_ HWND hwnd, _In_ LPCSTR lpszPhonebook)                                                                                                                                       |
| twui       | DWORD | RasCreatePhonebookEntryW(_In_ HWND hwnd, _In_ LPCWSTR lpszPhonebook)                                                                                                                                      |
| ssui       | DWORD | RasDeleteEntry(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry)                                                                                                                                        |
| aaui       | DWORD | RasDeleteEntryA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry)                                                                                                                                         |
| wwui       | DWORD | RasDeleteEntryW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry)                                                                                                                                       |
| ssuiui     | DWORD | RasDeleteSubEntry(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _In_ DWORD dwSubEntryId)                                                                                                            |
| aauiui     | DWORD | RasDeleteSubEntryA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _In_ DWORD dwSubEntryId)                                                                                                             |
| wwuiui     | DWORD | RasDeleteSubEntryW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _In_ DWORD dwSubEntryId)                                                                                                           |
| tstuiittui | DWORD | RasDial(_In_ LPRASDIALEXTENSIONS lpRasDialExtensions, _In_ LPCTSTR lpszPhonebook, _In_ LPRASDIALPARAMS lpRasDialParams, _In_ DWORD dwNotifierType, _In_ LPVOID lpvNotifier, _Out_ LPHRASCONN lphRasConn)  |
| tatuiittui | DWORD | RasDialA(_In_ LPRASDIALEXTENSIONS lpRasDialExtensions, _In_ LPCSTR lpszPhonebook, _In_ LPRASDIALPARAMS lpRasDialParams, _In_ DWORD dwNotifierType, _In_ LPVOID lpvNotifier, _Out_ LPHRASCONN lphRasConn)  |
| twtuiittui | DWORD | RasDialW(_In_ LPRASDIALEXTENSIONS lpRasDialExtensions, _In_ LPCWSTR lpszPhonebook, _In_ LPRASDIALPARAMS lpRasDialParams, _In_ DWORD dwNotifierType, _In_ LPVOID lpvNotifier, _Out_ LPHRASCONN lphRasConn) |
| tssui      | DWORD | RasEditPhonebookEntry(_In_ HWND hwnd, _In_ LPCTSTR                                                                                                                                                        |

|         |       |                                                                                                                                                                                                   |
|---------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |       | lpzPhonebook, _In_ LPCTSTR lpzEntryName)                                                                                                                                                          |
| taoui   | DWORD | RasEditPhonebookEntryA(_In_ HWND hwnd, _In_ LPCSTR lpzPhonebook, _In_ LPCSTR lpzEntryName)                                                                                                        |
| twwui   | DWORD | RasEditPhonebookEntryW(_In_ HWND hwnd, _In_ LPCWSTR lpzPhonebook, _In_ LPCWSTR lpzEntryName)                                                                                                      |
| tttui   | DWORD | RasEnumAutodialAddresses(_Inout_ LPTSTR *lppAddresses, _Inout_ LPDWORD lpdwcbAddresses, _Out_ LPDWORD lpdwcAddresses)                                                                             |
| tttui   | DWORD | RasEnumAutodialAddressesA(_Inout_ LPSTR *lppAddresses, _Inout_ LPDWORD lpdwcbAddresses, _Out_ LPDWORD lpdwcAddresses)                                                                             |
| tttui   | DWORD | RasEnumAutodialAddressesW(_Inout_ LPWSTR *lppAddresses, _Inout_ LPDWORD lpdwcbAddresses, _Out_ LPDWORD lpdwcAddresses)                                                                            |
| tttui   | DWORD | RasEnumConnections(_Inout_ LPRASCONN lprasconn, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcConnections)                                                                                               |
| tttui   | DWORD | RasEnumConnectionsA(_Inout_ LPRASCONN lprasconn, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcConnections)                                                                                              |
| tttui   | DWORD | RasEnumConnectionsW(_Inout_ LPRASCONN lprasconn, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcConnections)                                                                                              |
| tttui   | DWORD | RasEnumDevices(_In_ LPRASDEVINFO lpRasDevInfo, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcDevices)                                                                                                    |
| tttui   | DWORD | RasEnumDevicesA(_In_ LPRASDEVINFO lpRasDevInfo, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcDevices)                                                                                                   |
| tttui   | DWORD | RasEnumDevicesW(_In_ LPRASDEVINFO lpRasDevInfo, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcDevices)                                                                                                   |
| ssttui  | DWORD | RasEnumEntries(_In_ LPCTSTR reserved, _In_ LPCTSTR lpzPhonebook, _Inout_ LPRASENTRYNAME lprasentryname, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcEntries)                                           |
| aatttui | DWORD | RasEnumEntriesA(_In_ LPCSTR reserved, _In_ LPCSTR lpzPhonebook, _Inout_ LPRASENTRYNAME lprasentryname, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcEntries)                                            |
| wwtttui | DWORD | RasEnumEntriesW(_In_ LPCWSTR reserved, _In_ LPCWSTR lpzPhonebook, _Inout_ LPRASENTRYNAME lprasentryname, _Inout_ LPDWORD lpcb, _Out_ LPDWORD lpcEntries)                                          |
| ti      | VOID  | RasFreeEapUserIdentity(_In_ LPRASEAPUSERIDENTITY pRasEapUserIdentity)                                                                                                                             |
| ti      | VOID  | RasFreeEapUserIdentityA(_In_ LPRASEAPUSERIDENTITY pRasEapUserIdentity)                                                                                                                            |
| ti      | VOID  | RasFreeEapUserIdentityW(_In_ LPRASEAPUSERIDENTITY pRasEapUserIdentity)                                                                                                                            |
| stttui  | DWORD | RasGetAutodialAddress(_In_ LPCTSTR lpzAddress, _In_ LPDWORD lpdwReserved, _Inout_ LPRASAUTODIALENTY lpAutoDialEntries, _Inout_ LPDWORD lpdwcbAutoDialEntries, _Out_ LPDWORD lpdwcAutoDialEntries) |
|         |       |                                                                                                                                                                                                   |

|        |       |                                                                                                                                                                                                     |
|--------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| atttui | DWORD | RasGetAutodialAddressA(_In_ LPCSTR lpszAddress, _In_ LPDWORD lpdwReserved, _Inout_ LPRASAUTODIALENTY lpAutoDialEntries, _Inout_ LPDWORD lpdwcbAutoDialEntries, _Out_ LPDWORD lpdwcAutoDialEntries)  |
| wtttui | DWORD | RasGetAutodialAddressW(_In_ LPCWSTR lpszAddress, _In_ LPDWORD lpdwReserved, _Inout_ LPRASAUTODIALENTY lpAutoDialEntries, _Inout_ LPDWORD lpdwcbAutoDialEntries, _Out_ LPDWORD lpdwcAutoDialEntries) |
| uitui  | DWORD | RasGetAutodialEnable(_In_ DWORD dwDialingLocation, _Out_ LPBOOL lpfEnabled)                                                                                                                         |
| uitui  | DWORD | RasGetAutodialEnableA(_In_ DWORD dwDialingLocation, _Out_ LPBOOL lpfEnabled)                                                                                                                        |
| uitui  | DWORD | RasGetAutodialEnableW(_In_ DWORD dwDialingLocation, _Out_ LPBOOL lpfEnabled)                                                                                                                        |
| uittui | DWORD | RasGetAutodialParam(_In_ DWORD dwKey, _Out_ LPVOID lpvValue, _Inout_ LPDWORD lpdwcbValue)                                                                                                           |
| uittui | DWORD | RasGetAutodialParamA(_In_ DWORD dwKey, _Out_ LPVOID lpvValue, _Inout_ LPDWORD lpdwcbValue)                                                                                                          |
| uittui | DWORD | RasGetAutodialParamW(_In_ DWORD dwKey, _Out_ LPVOID lpvValue, _Inout_ LPDWORD lpdwcbValue)                                                                                                          |
| ttui   | DWORD | RasGetConnectionStatistics(_In_ HRASCONN hRasConn, _Inout_ RAS_STATS *lpStatistics)                                                                                                                 |
| ttui   | DWORD | RasGetConnectStatus(_In_ HRASCONN hrasconn, _Inout_ LPRASCONNSTATUS lprasconnstatus)                                                                                                                |
| ttui   | DWORD | RasGetConnectStatusA(_In_ HRASCONN hrasconn, _Inout_ LPRASCONNSTATUS lprasconnstatus)                                                                                                               |
| ttui   | DWORD | RasGetConnectStatusW(_In_ HRASCONN hrasconn, _Inout_ LPRASCONNSTATUS lprasconnstatus)                                                                                                               |
| ttui   | DWORD | RasGetCountryInfo(_Inout_ LPRASCTRYINFO lpRasCtryInfo, _Inout_ LPDWORD lpdwSize)                                                                                                                    |
| ttui   | DWORD | RasGetCountryInfoA(_Inout_ LPRASCTRYINFO lpRasCtryInfo, _Inout_ LPDWORD lpdwSize)                                                                                                                   |
| ttui   | DWORD | RasGetCountryInfoW(_Inout_ LPRASCTRYINFO lpRasCtryInfo, _Inout_ LPDWORD lpdwSize)                                                                                                                   |
| sstui  | DWORD | RasGetCredentialA(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _Inout_ LPRASCREDENTIALS lpCredentials)                                                                                       |
| aatui  | DWORD | RasGetCredentialsA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _Inout_ LPRASCREDENTIALS lpCredentials)                                                                                        |
| wwtui  | DWORD | RasGetCredentialsW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _Inout_ LPRASCREDENTIALS lpCredentials)                                                                                      |
| wwtui  | DWORD | RasGetCustomAuthData(_In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _Out_ BYTE *pbCustomAuthData, _Inout_ DWORD *pdwSizeofCustomAuthData)                                                        |

|           |       |                                                                                                                                                                                                              |
|-----------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aattui    | DWORD | RasGetCustomAuthDataA(_In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _Out_ BYTE *pbCustomAuthData, _Inout_ DWORD *pdwSizeofCustomAuthData)                                                                |
| wwttui    | DWORD | RasGetCustomAuthDataW(_In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _Out_ BYTE *pbCustomAuthData, _Inout_ DWORD *pdwSizeofCustomAuthData)                                                                |
| tsstui    | DWORD | RasGetEapUserData(_In_ HANDLE hToken, _In_ LPCTSTR pszPhonebook, _In_ LPCTSTR pszEntry, _Out_ BYTE *pbEapData, _Inout_ DWORD *pdwSizeofEapData)                                                              |
| taattui   | DWORD | RasGetEapUserDataA(_In_ HANDLE hToken, _In_ LPCSTR pszPhonebook, _In_ LPCSTR pszEntry, _Out_ BYTE *pbEapData, _Inout_ DWORD *pdwSizeofEapData)                                                               |
| twwttui   | DWORD | RasGetEapUserDataW(_In_ HANDLE hToken, _In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _Out_ BYTE *pbEapData, _Inout_ DWORD *pdwSizeofEapData)                                                             |
| aauiittui | DWORD | RasGetEapUserIdentity(_In_ LPCSTR pszPhonebook, _In_ LPCSTR pszEntry, _In_ DWORD dwFlags, _In_ HWND hwnd, _Out_ LPRASEAPUSERIDENTITY *ppRasEapUserIdentity)                                                  |
| aauiittui | DWORD | RasGetEapUserIdentityA(_In_ LPCSTR pszPhonebook, _In_ LPCSTR pszEntry, _In_ DWORD dwFlags, _In_ HWND hwnd, _Out_ LPRASEAPUSERIDENTITY *ppRasEapUserIdentity)                                                 |
| aauiittui | DWORD | RasGetEapUserIdentityW(_In_ LPCSTR pszPhonebook, _In_ LPCSTR pszEntry, _In_ DWORD dwFlags, _In_ HWND hwnd, _Out_ LPRASEAPUSERIDENTITY *ppRasEapUserIdentity)                                                 |
| sttui     | DWORD | RasGetEntryDialParams(_In_ LPCTSTR lpszPhonebook, _Inout_ LPRASDIALPARAMS lprasdialparams, _Out_ LPBOOL lpfPassword)                                                                                         |
| attui     | DWORD | RasGetEntryDialParamsA(_In_ LPCSTR lpszPhonebook, _Inout_ LPRASDIALPARAMS lprasdialparams, _Out_ LPBOOL lpfPassword)                                                                                         |
| wttui     | DWORD | RasGetEntryDialParamsW(_In_ LPCWSTR lpszPhonebook, _Inout_ LPRASDIALPARAMS lprasdialparams, _Out_ LPBOOL lpfPassword)                                                                                        |
| sstttui   | DWORD | RasGetEntryProperties(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _Inout_ LPRASENTRY lpRasEntry, _Inout_ LPDWORD lpdwEntryInfoSize, _Out_ LPBYTE lpbDeviceInfo, _Inout_ LPDWORD lpdwDeviceInfoSize)  |
| aattttui  | DWORD | RasGetEntryPropertiesA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _Inout_ LPRASENTRY lpRasEntry, _Inout_ LPDWORD lpdwEntryInfoSize, _Out_ LPBYTE lpbDeviceInfo, _Inout_ LPDWORD lpdwDeviceInfoSize)   |
| wwttttui  | DWORD | RasGetEntryPropertiesW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _Inout_ LPRASENTRY lpRasEntry, _Inout_ LPDWORD lpdwEntryInfoSize, _Out_ LPBYTE lpbDeviceInfo, _Inout_ LPDWORD lpdwDeviceInfoSize) |
| uisuiui   | DWORD | RasGetErrorString(_In_ UINT uErrorValue, _Out_ LPTSTR lpszErrorString, _In_ DWORD cBufSize)                                                                                                                  |
|           |       |                                                                                                                                                                                                              |

|            |       |                                                                                                                                                                                                                               |
|------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiauiui    | DWORD | RasGetErrorStringA(_In_ UINT uErrorValue, _Out_ LPSTR lpszErrorString, _In_ DWORD cBufSize)                                                                                                                                   |
| uiwuiui    | DWORD | RasGetErrorStringW(_In_ UINT uErrorValue, _Out_ LPWSTR lpszErrorString, _In_ DWORD cBufSize)                                                                                                                                  |
| tuitui     | DWORD | RasGetLinkStatistics(_In_ HRASCONN hRasConn, _In_ DWORD dwSubEntry, _Inout_ RAS_STATS *lpStatistics)                                                                                                                          |
| ttui       | DWORD | rasgetnapstatus(_In_ HRASCONN hRasConn, _Inout_ LPRASNAPSTATE pNapState)                                                                                                                                                      |
| tittui     | DWORD | RasGetProjectionInfo(_In_ HRASCONN hrasconn, _In_ RASPROJECTION rasprojection, _Out_ LPVOID lpprojection, _Inout_ LPDWORD lpcb)                                                                                               |
| tittui     | DWORD | RasGetProjectionInfoA(_In_ HRASCONN hrasconn, _In_ RASPROJECTION rasprojection, _Out_ LPVOID lpprojection, _Inout_ LPDWORD lpcb)                                                                                              |
| tttui      | DWORD | RasGetProjectionInfoEx(_In_ HRASCONN Hrasconn, _Inout_ PRAS_PROJECTION_INFO pRasProjection, _Inout_ LPDWORD lpdwSize)                                                                                                         |
| tittui     | DWORD | RasGetProjectionInfoW(_In_ HRASCONN hrasconn, _In_ RASPROJECTION rasprojection, _Out_ LPVOID lpprojection, _Inout_ LPDWORD lpcb)                                                                                              |
| tuitui     | DWORD | RasGetSubEntryHandle(_In_ HRASCONN hRasConn, _In_ DWORD dwSubEntry, _Out_ LPHRASCONN lphRasConn)                                                                                                                              |
| tuitui     | DWORD | RasGetSubEntryHandleA(_In_ HRASCONN hRasConn, _In_ DWORD dwSubEntry, _Out_ LPHRASCONN lphRasConn)                                                                                                                             |
| tuitui     | DWORD | RasGetSubEntryHandleW(_In_ HRASCONN hRasConn, _In_ DWORD dwSubEntry, _Out_ LPHRASCONN lphRasConn)                                                                                                                             |
| ssuitttui  | DWORD | RasGetSubEntryProperties(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _In_ DWORD dwSubEntry, _Inout_ LPRASSUBENTRY lpRasSubEntry, _Inout_ LPDWORD lpdwcb, _In_ LPBYTE lpbDeviceConfig, _In_ LPDWORD lpcbDeviceConfig)  |
| aauiitttui | DWORD | RasGetSubEntryPropertiesA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _In_ DWORD dwSubEntry, _Inout_ LPRASSUBENTRY lpRasSubEntry, _Inout_ LPDWORD lpdwcb, _In_ LPBYTE lpbDeviceConfig, _In_ LPDWORD lpcbDeviceConfig)   |
| wwuitttui  | DWORD | RasGetSubEntryPropertiesW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _In_ DWORD dwSubEntry, _Inout_ LPRASSUBENTRY lpRasSubEntry, _Inout_ LPDWORD lpdwcb, _In_ LPBYTE lpbDeviceConfig, _In_ LPDWORD lpcbDeviceConfig) |
| tui        | DWORD | RasHangUp(_In_ HRASCONN hRasConn)                                                                                                                                                                                             |
| tui        | DWORD | RasHangUpA(_In_ HRASCONN hRasConn)                                                                                                                                                                                            |
| tui        | DWORD | RasHangUpW(_In_ HRASCONN hRasConn)                                                                                                                                                                                            |
| tuittui    | DWORD | RasInvokeEapUI(_In_ HRASCONN hRasConn, _In_ DWORD dwSubEntry, _In_ LPRASDIALEXTENSIONS lpExtensions, _In_ HWND hwnd)                                                                                                          |
|            |       | RasRenameEntry(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR                                                                                                                                                                       |

|            |       |                                                                                                                                                                                  |
|------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sssui      | DWORD | lpszOldEntry, _In_ LPCTSTR lpszNewEntry)                                                                                                                                         |
| aaaii      | DWORD | RasRenameEntryA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszOldEntry, _In_ LPCSTR lpszNewEntry)                                                                                   |
| wwwui      | DWORD | RasRenameEntryW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszOldEntry, _In_ LPCWSTR lpszNewEntry)                                                                                |
| suituiiui  | DWORD | RasSetAutodialAddress(_In_ LPCTSTR lpszAddress, _In_ DWORD dwReserved, _In_ LPRASAUTODIALENTY lpAutoDialEntries, _In_ DWORD dwcbAutoDialEntries, _In_ DWORD dwcAutoDialEntries)  |
| aituiiuii  | DWORD | RasSetAutodialAddressA(_In_ LPCSTR lpszAddress, _In_ DWORD dwReserved, _In_ LPRASAUTODIALENTY lpAutoDialEntries, _In_ DWORD dwcbAutoDialEntries, _In_ DWORD dwcAutoDialEntries)  |
| wuituiiuii | DWORD | RasSetAutodialAddressW(_In_ LPCWSTR lpszAddress, _In_ DWORD dwReserved, _In_ LPRASAUTODIALENTY lpAutoDialEntries, _In_ DWORD dwcbAutoDialEntries, _In_ DWORD dwcAutoDialEntries) |
| uiiui      | DWORD | RasSetAutodialEnable(_In_ DWORD dwDialingLocation, _In_ BOOL fEnabled)                                                                                                           |
| uiiui      | DWORD | RasSetAutodialEnableA(_In_ DWORD dwDialingLocation, _In_ BOOL fEnabled)                                                                                                          |
| uiiui      | DWORD | RasSetAutodialEnableW(_In_ DWORD dwDialingLocation, _In_ BOOL fEnabled)                                                                                                          |
| uituiui    | DWORD | RasSetAutodialParam(_In_ DWORD dwKey, _Out_ LPVOID lpvValue, _In_ DWORD dwcbValue)                                                                                               |
| uituiui    | DWORD | RasSetAutodialParamA(_In_ DWORD dwKey, _Out_ LPVOID lpvValue, _In_ DWORD dwcbValue)                                                                                              |
| uituiui    | DWORD | RasSetAutodialParamW(_In_ DWORD dwKey, _Out_ LPVOID lpvValue, _In_ DWORD dwcbValue)                                                                                              |
| ssuii      | DWORD | RasSetCredentials(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _In_ LPRASCREDENTIALS lpCredentials, _In_ BOOL fClearCredentials)                                          |
| aatuii     | DWORD | RasSetCredentialsA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _In_ LPRASCREDENTIALS lpCredentials, _In_ BOOL fClearCredentials)                                           |
| wwtuii     | DWORD | RasSetCredentialsW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _In_ LPRASCREDENTIALS lpCredentials, _In_ BOOL fClearCredentials)                                         |
| wwtuiui    | DWORD | RasSetCustomAuthData(_In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _In_ BYTE *pbCustomAuthData, _In_ DWORD dwSizeofCustomAuthData)                                           |
| aatuiui    | DWORD | RasSetCustomAuthDataA(_In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _In_ BYTE *pbCustomAuthData, _In_ DWORD dwSizeofCustomAuthData)                                          |
| wwtuiui    | DWORD | RasSetCustomAuthDataW(_In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _In_ BYTE *pbCustomAuthData, _In_ DWORD dwSizeofCustomAuthData)                                          |

|              |       |                                                                                                                                                                                                                              |
|--------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsstuiui     | DWORD | RasSetEapUsesData(_In_ HANDLE hToken, _In_ LPCTSTR pszPhonebook, _In_ LPCTSTR pszEntry, _In_ BYTE *pbEapData, _In_ DWORD dwSizeofEapData)                                                                                    |
| taatuiui     | DWORD | RasSetEapUserDataA(_In_ HANDLE hToken, _In_ LPCSTR pszPhonebook, _In_ LPCSTR pszEntry, _In_ BYTE *pbEapData, _In_ DWORD dwSizeofEapData)                                                                                     |
| twwtuiui     | DWORD | RasSetEapUserDataW(_In_ HANDLE hToken, _In_ LPCWSTR pszPhonebook, _In_ LPCWSTR pszEntry, _In_ BYTE *pbEapData, _In_ DWORD dwSizeofEapData)                                                                                   |
| stuii        | DWORD | RasSetEntryDialParams(_In_ LPCTSTR lpszPhonebook, _In_ LPRASDIALPARAMS lprasdialparams, _In_ BOOL fRemovePassword)                                                                                                           |
| atiui        | DWORD | RasSetEntryDialParamsA(_In_ LPCSTR lpszPhonebook, _In_ LPRASDIALPARAMS lprasdialparams, _In_ BOOL fRemovePassword)                                                                                                           |
| wtiui        | DWORD | RasSetEntryDialParamsW(_In_ LPCWSTR lpszPhonebook, _In_ LPRASDIALPARAMS lprasdialparams, _In_ BOOL fRemovePassword)                                                                                                          |
| ssuituiui    | DWORD | RasSetEntryProperties(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _In_ LPRASENTRY lpRasEntry, _In_ DWORD dwEntryInfoSize, _In_ LPBYTE lpbDeviceInfo, _In_ DWORD dwDeviceInfoSize)                                    |
| aatuituiui   | DWORD | RasSetEntryPropertiesA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _In_ LPRASENTRY lpRasEntry, _In_ DWORD dwEntryInfoSize, _In_ LPBYTE lpbDeviceInfo, _In_ DWORD dwDeviceInfoSize)                                     |
| wwtuituiui   | DWORD | RasSetEntryPropertiesW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _In_ LPRASENTRY lpRasEntry, _In_ DWORD dwEntryInfoSize, _In_ LPBYTE lpbDeviceInfo, _In_ DWORD dwDeviceInfoSize)                                   |
| ssuituituiui | DWORD | RasSetSubEntryProperties(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry, _In_ DWORD dwSubEntry, _In_ LPRASSUBENTRY lpRasSubEntry, _In_ DWORD dwcbRasSubEntry, _In_ LPBYTE lpbDeviceConfig, _In_ DWORD dwcbDeviceConfig)  |
| aauituituiui | DWORD | RasSetSubEntryPropertiesA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry, _In_ DWORD dwSubEntry, _In_ LPRASSUBENTRY lpRasSubEntry, _In_ DWORD dwcbRasSubEntry, _In_ LPBYTE lpbDeviceConfig, _In_ DWORD dwcbDeviceConfig)   |
| wwuituituiui | DWORD | RasSetSubEntryPropertiesW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR lpszEntry, _In_ DWORD dwSubEntry, _In_ LPRASSUBENTRY lpRasSubEntry, _In_ DWORD dwcbRasSubEntry, _In_ LPBYTE lpbDeviceConfig, _In_ DWORD dwcbDeviceConfig) |
| ttui         | DWORD | RasUpdateConnection(_In_ HRASCONN hrasconn, _In_ LPRASUPDATECONN lprasupdateconn)                                                                                                                                            |
| ssui         | DWORD | RasValidateEntryName(_In_ LPCTSTR lpszPhonebook, _In_ LPCTSTR lpszEntry)                                                                                                                                                     |
| aaui         | DWORD | RasValidateEntryNameA(_In_ LPCSTR lpszPhonebook, _In_ LPCSTR lpszEntry)                                                                                                                                                      |
|              |       | RasValidateEntryNameW(_In_ LPCWSTR lpszPhonebook, _In_ LPCWSTR                                                                                                                                                               |

wwui

DWORD

lpzEntry)

## Rasdlg.dll

|       |      |                                                                                                                      |
|-------|------|----------------------------------------------------------------------------------------------------------------------|
| sssti | BOOL | RasDialDlg(_In_ LPTSTR lpszPhonebook, _In_ LPTSTR lpszEntry, _In_ LPTSTR lpszPhoneNumber, _In_ LPRASDIALDLG lpInfo)  |
| aaati | BOOL | RasDialDlgA(_In_ LPSTR lpszPhonebook, _In_ LPSTR lpszEntry, _In_ LPSTR lpszPhoneNumber, _In_ LPRASDIALDLG lpInfo)    |
| wwwti | BOOL | RasDialDlgW(_In_ LPWSTR lpszPhonebook, _In_ LPWSTR lpszEntry, _In_ LPWSTR lpszPhoneNumber, _In_ LPRASDIALDLG lpInfo) |
| ssti  | BOOL | RasEntryDlg(_In_ LPTSTR lpszPhonebook, _In_ LPTSTR lpszEntry, _In_ LPRASENTRYDLG lpInfo)                             |
| aati  | BOOL | RasEntryDlgA(_In_ LPSTR lpszPhonebook, _In_ LPSTR lpszEntry, _In_ LPRASENTRYDLG lpInfo)                              |
| wwti  | BOOL | RasEntryDlgW(_In_ LPWSTR lpszPhonebook, _In_ LPWSTR lpszEntry, _In_ LPRASENTRYDLG lpInfo)                            |
| ssti  | BOOL | RasPhonebookDlg(_In_ LPTSTR lpszPhonebook, _In_ LPTSTR lpszEntry, _Inout_ LPRASPBDLG lpInfo)                         |
| aati  | BOOL | RasPhonebookDlgA(_In_ LPSTR lpszPhonebook, _In_ LPSTR lpszEntry, _Inout_ LPRASPBDLG lpInfo)                          |
| wwti  | BOOL | RasPhonebookDlgW(_In_ LPWSTR lpszPhonebook, _In_ LPWSTR lpszEntry, _Inout_ LPRASPBDLG lpInfo)                        |

## Rasman.dll

|         |       |                                                                                                                                                   |
|---------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| ttui    | DWORD | <a href="#">RasSecurityDialogGetInfo</a> (_In_ HPORT hPort, _In_ RAS_SECURITY_INFO *pBuffer)                                                      |
| ttuitui | DWORD | <a href="#">RasSecurityDialogReceive</a> (_In_ HPORT hPort, _In_ PBYTE pBuffer, _In_ PWORD pBufferLength, _In_ DWORD Timeout, _In_ HANDLE hEvent) |
| ttuhui  | DWORD | <a href="#">RasSecurityDialogSend</a> (_In_ HPORT hPort, _In_ PBYTE pBuffer, _In_ WORD BufferLength)                                              |

## Shell32.dll

|              |         |                                                                                                                                                                                                                                                                                  |
|--------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti       | HRESULT | <code>AssocCreateForClass</code> (_In_ const ASSOCIATIONELEMENT *rgClasses, _In_ ULONG cClasses, _In_ REFIID riid, _Out_ void **ppv)                                                                                                                                             |
| tttti        | HRESULT | <code>AssocGetDetailsOfPropKey</code> (_In_ IShellFolder *psf, _In_ PCUITEMID_CHILD pidl, _In_ PROPERTYKEY *pkey, _Out_ VARIANT *pv, _Out_ BOOL *pfFoundPropKey)                                                                                                                 |
| ttuitttuitti | HRESULT | <code>CDefFolderMenu_Create2</code> (_In_opt_ PCIDLIST_ABSOLUTE pidlFolder, _In_opt_ HWND hwnd, UINT cidl, _In_opt_ PCUITEMID_CHILD_ARRAY *apidl, _In_opt_ IShellFolder *psf, _In_opt_ LPFNDFM_CALLBACK lpf, UINT nKeys, _In_opt_ const HKEY *ahkeys, _Out_ IContextMenu **ppcm) |
| tuitti       | HRESULT | <code>CIDLData_CreateFromIDArray</code> (_In_ PCIDLIST_ABSOLUTE pidlFolder, _In_ UINT cidl, _In_ PCUIDLIST_RELATIVE_ARRAY apidl, _Out_ IDataObject **ppdtobj)                                                                                                                    |
| wtt          | LPWSTR* | <code>CommandLineToArgvW</code> (_In_ LPCWSTR lpCmdLine, _Out_ int *pNumArgs)                                                                                                                                                                                                    |
| ttti         | BOOL    | <code>DAD_AutoScroll</code> (_In_ HWND hwnd, _In_ AUTO_SCROLL_DATA *pad, _In_ const POINT *pptNow)                                                                                                                                                                               |
| ti6i         | BOOL    | <code>DAD_DragEnterEx</code> (HWND hwndTarget, const POINT ptStart)                                                                                                                                                                                                              |
| ti6ti        | BOOL    | <code>DAD_DragEnterEx2</code> (_In_ HWND hwndTarget, const POINT ptStart, _In_opt_ IDataObject *pdtObject)                                                                                                                                                                       |
| i            | BOOL    | <code>DAD_DragLeave</code> (void)                                                                                                                                                                                                                                                |
| i6i          | BOOL    | <code>DAD_DragMove</code> (POINT pt)                                                                                                                                                                                                                                             |
| ttti         | BOOL    | <code>DAD_SetDragImage</code> (HIMAGELIST him, POINT *pptOffset)                                                                                                                                                                                                                 |
| ii           | BOOL    | <code>DAD_ShowDragImage</code> (BOOL fShow)                                                                                                                                                                                                                                      |
| ti           | HRESULT | <code>DllGetVersion</code> (DLLVERSIONINFO *pdvi)                                                                                                                                                                                                                                |
| suiui        | DWORD   | <code>DoEnvironmentSubst</code> (_Inout_ LPTSTR pszSrc, UINT cchSrc)                                                                                                                                                                                                             |
| auui         | DWORD   | <code>DoEnvironmentSubstA</code> (_Inout_ LPSTR pszSrc, UINT cchSrc)                                                                                                                                                                                                             |
| wuiui        | DWORD   | <code>DoEnvironmentSubstW</code> (_Inout_ LPWSTR pszSrc, UINT cchSrc)                                                                                                                                                                                                            |
| tii          | VOID    | <code>DragAcceptFiles</code> (HWND hWnd, BOOL fAccept)                                                                                                                                                                                                                           |
| ti           | VOID    | <code>DragFinish</code> (HDROP hDrop)                                                                                                                                                                                                                                            |
| tuisuiui     | UINT    | <code>DragQueryFile</code> (_In_ HDROP hDrop, _In_ UINT iFile, _Out_ LPTSTR lpszFile, UINT cch)                                                                                                                                                                                  |
| tuiuiui      | UINT    | <code>DragQueryFileA</code> (_In_ HDROP hDrop, _In_ UINT iFile, _Out_ LPSTR lpszFile, UINT cch)                                                                                                                                                                                  |

|          |           |                                                                                                                                          |
|----------|-----------|------------------------------------------------------------------------------------------------------------------------------------------|
| tuiwuiui | UINT      | DragQueryFileW(_In_ HDROP hDrop, _In_ UINT iFile, _Out_ LPWSTR lpszFile, UINT cch)                                                       |
| tti      | BOOL      | DragQueryPoint(_In_ HDROP hDrop, _Out_ POINT *lppt)                                                                                      |
| ii       | int       | DriveType(_In_ int iDrive)                                                                                                               |
| ttt      | HICON     | DuplicateIcon(_Reserved_ HINSTANCE hInst, _In_ HICON hIcon)                                                                              |
| tsst     | HICON     | ExtractAssociatedIcon(_Reserved_ HINSTANCE hInst, _Inout_ LPTSTR lpIconPath, _Inout_ WORD *lpIcon)                                       |
| tatt     | HICON     | ExtractAssociatedIconA(_Reserved_ HINSTANCE hInst, _Inout_ LPSTR lpIconPath, _Inout_ WORD *lpIcon)                                       |
| tsstt    | HICON     | ExtractAssociatedIconEx(_Reserved_ HINSTANCE hInst, _Inout_ LPTSTR lpIconPath, _Inout_ LPWORD lpIconIndex, _Inout_ LPWORD lpIconId)      |
| tattt    | HICON     | ExtractAssociatedIconExA(_Reserved_ HINSTANCE hInst, _Inout_ LPSTR lpIconPath, _Inout_ LPWORD lpIconIndex, _Inout_ LPWORD lpIconId)      |
| twttt    | HICON     | ExtractAssociatedIconExW(_Reserved_ HINSTANCE hInst, _Inout_ LPWSTR lpIconPath, _Inout_ LPWORD lpIconIndex, _Inout_ LPWORD lpIconId)     |
| twtt     | HICON     | ExtractAssociatedIconW(_Reserved_ HINSTANCE hInst, _Inout_ LPWSTR lpIconPath, _Inout_ WORD *lpIcon)                                      |
| tsuit    | HICON     | ExtractIcon(_Reserved_ HINSTANCE hInst, _In_ LPCTSTR lpszExeFileName, _In_ UINT nIconIndex)                                              |
| tauit    | HICON     | ExtractIconA(_Reserved_ HINSTANCE hInst, _In_ LPCSTR lpszExeFileName, _In_ UINT nIconIndex)                                              |
| sittuiui | UINT      | ExtractIconEx(_In_ LPCTSTR lpszFile, _In_ int nIconIndex, _Out_opt_ HICON *phiconLarge, _Out_opt_ HICON *phiconSmall, _In_ UINT nIcons)  |
| aittuiui | UINT      | ExtractIconExA(_In_ LPCSTR lpszFile, _In_ int nIconIndex, _Out_opt_ HICON *phiconLarge, _Out_opt_ HICON *phiconSmall, _In_ UINT nIcons)  |
| wittuiui | UINT      | ExtractIconExW(_In_ LPCWSTR lpszFile, _In_ int nIconIndex, _Out_opt_ HICON *phiconLarge, _Out_opt_ HICON *phiconSmall, _In_ UINT nIcons) |
| twuit    | HICON     | ExtractIconW(_Reserved_ HINSTANCE hInst, _In_ LPCWSTR lpszExeFileName, _In_ UINT nIconIndex)                                             |
| ssst     | HINSTANCE | FindExecutable(_In_ LPCTSTR lpFile, _In_opt_ LPCTSTR lpDirectory, _Out_ LPTSTR lpResult)                                                 |
| aaat     | HINSTANCE | FindExecutableA(_In_ LPCSTR lpFile, _In_opt_ LPCSTR lpDirectory, _Out_ LPSTR lpResult)                                                   |
| wwwt     | HINSTANCE | FindExecutableW(_In_ LPCWSTR lpFile, _In_opt_ LPCWSTR lpDirectory, _Out_ LPWSTR lpResult)                                                |
|          |           |                                                                                                                                          |

|           |                   |                                                                                                                                                                                                   |
|-----------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wi        | HRESULT           | GetCurrentProcessExplicitAppUserModelID(_Out_ PWSTR *AppID)                                                                                                                                       |
| ttuiwwwwi | BOOL              | GetFileNameFromBrowse(_In_opt_ HWND hwnd, _Inout_ PWSTR pszFilePath, UINT cchFilePath, _In_opt_ PCWSTR pszWorkingDir, _In_ PCWSTR pszDefExt, _In_opt_ PCWSTR pszFilters, _In_opt_ PCWSTR szTitle) |
| ttit      | PIDLIST_RELATIVE  | ILAppendID(_In_opt_ PIDLIST_RELATIVE pidl, _In_ LPSHITEMID pmkid, BOOL fAppend)                                                                                                                   |
| tt        | PIDLIST_RELATIVE  | ILClose(_In_ PCUIDLIST_RELATIVE pidl)                                                                                                                                                             |
| tt        | PITEMID_CHILD     | ILCloseFirst(_In_ PCUIDLIST_RELATIVE pidl)                                                                                                                                                        |
| ttt       | PIDLIST_ABSOLUTE  | ILCombine(_In_ PCIDLIST_ABSOLUTE pidl1, _In_ PCUIDLIST_RELATIVE pidl2)                                                                                                                            |
| st        | PIDLIST_ABSOLUTE  | ILCreateFromPath(_In_ PCTSTR pszPath)                                                                                                                                                             |
| at        | PIDLIST_ABSOLUTE  | ILCreateFromPathA(_In_ PCTSTR pszPath)                                                                                                                                                            |
| wt        | PIDLIST_ABSOLUTE  | ILCreateFromPathW(_In_ PCTSTR pszPath)                                                                                                                                                            |
| ttt       | PUIDLIST_RELATIVE | ILFindChild(_In_ PCIDLIST_ABSOLUTE pidlParent, _In_ PCIDLIST_ABSOLUTE pidlChild)                                                                                                                  |
| tt        | PUITEMID_CHILD    | ILFindLastID(_In_ PCUIDLIST_RELATIVE pidl)                                                                                                                                                        |
| ti        | VOID              | ILFirst(_In_ PIDLIST_RELATIVE pidl)                                                                                                                                                               |
| tt        | PUIDLIST_RELATIVE | ILGetNext(_In_opt_ PCUIDLIST_RELATIVE pidl)                                                                                                                                                       |
| tui       | UINT              | ILGetSize(_In_opt_ PCUIDLIST_RELATIVE pidl)                                                                                                                                                       |
| tti       | BOOL              | ILIsEqual(_In_ PCIDLIST_ABSOLUTE pidl1, _In_ PCIDLIST_ABSOLUTE pidl2)                                                                                                                             |
| ttii      | BOOL              | ILIsParent(_In_ PCIDLIST_ABSOLUTE pidl1, _In_ PCIDLIST_ABSOLUTE pidl2, _In_ BOOL fImmediate)                                                                                                      |
| tti       | HRESULT           | ILLoadFromStreamEx(_In_ IStream *pstm, _Out_ PIDLIST_RELATIVE *pidl)                                                                                                                              |
| ti        | BOOL              | ILRemoveLastID(_Inout_opt_ PUIDLIST_RELATIVE pidl)                                                                                                                                                |
| tti       | HRESULT           | ILSaveToStream(_In_ IStream *pstm, _In_ PCUIDLIST_RELATIVE pidl)                                                                                                                                  |
| i         | BOOL              | InitNetworkAddressControl(void)                                                                                                                                                                   |
| ii        | int               | IsNetDrive(_In_ int iDrive)                                                                                                                                                                       |
| i         | BOOL              | IsUserAnAdmin(void)                                                                                                                                                                               |
| twwuit    | IStream*          | OpenRegStream(_In_ HKEY hkey, _In_opt_ PCWSTR pszSubkey, _In_opt_ PCWSTR pszValue, DWORD grfMode)                                                                                                 |
| wti       | int               | PathCleanupSpec(_In_opt_ PCWSTR pszDir, _Inout_ PWSTR pszSpec)                                                                                                                                    |
| ti        | VOID              | PathGetShortPath(_Inout_ PWSTR pszLongPath)                                                                                                                                                       |
|           |                   |                                                                                                                                                                                                   |

|         |         |                                                                                                                                           |
|---------|---------|-------------------------------------------------------------------------------------------------------------------------------------------|
| wi      | BOOL    | PathIsExe(_In_ PCWSTR szfile)                                                                                                             |
| suii    | BOOL    | PathIsSlow(_In_ LPCTSTR pszFile, DWORD dwFileAttr)                                                                                        |
| auii    | BOOL    | PathIsSlowA(_In_ LPCSTR pszFile, DWORD dwFileAttr)                                                                                        |
| wuii    | BOOL    | PathIsSlowW(_In_ LPCWSTR pszFile, DWORD dwFileAttr)                                                                                       |
| tuiwwi  | BOOL    | PathMakeUniqueName(_Out_ PWSTR pszUniqueName, UINT cchMax, _In_ PCWSTR pszTemplate, _In_opt_ PCWSTR pszLongPlate, _In_opt_ PCWSTR pszDir) |
| twuii   | int     | PathResolve(_Inout_ PWSTR pszPath, _In_opt_ PZPCWSTR dirs, UINT fFlags)                                                                   |
| twwwi   | BOOL    | PathYetAnotherMakeUniqueName(_Out_ PWSTR pszUniqueName, _In_ PCWSTR pszPath, _In_opt_ PCWSTR pszShort, _In_opt_ PCWSTR pszFileSpec)       |
| ttuiti  | int     | PickIconDlg(_In_opt_ HWND hwnd, _Inout_ PWSTR pszIconPath, UINT cchIconPath, _Inout_opt_ int *piIconIndex)                                |
| tuii    | int     | PifMgr_CloseProperties(_In_ HANDLE hProps, _In_ UINT fOpt)                                                                                |
| tatiuii | int     | PifMgr_GetProperties(_In_opt_ HANDLE hProps, _In_opt_ PCSTR pszGroup, _Out_opt_ void *lpProps, int cbProps, UINT fOpt)                    |
| wwuiuit | HANDLE  | PifMgr_OpenProperties(_In_ PCWSTR pszApp, _In_opt_ PCWSTR lpszPIF, UINT hInf, UINT fOpt)                                                  |
| tatiuii | int     | PifMgr_SetProperties(_In_opt_ HANDLE hProps, _In_opt_ PCSTR pszGroup, _In_ const void *lpProps, int cbProps, UINT fOpt)                   |
| tii     | BOOL    | ReadCabinetState(_Out_ CABINETSTATE *pcs, _In_ int cLength)                                                                               |
| iii     | int     | RealDriveType(_In_ int iDrive, _Reserved_ BOOL fOKToHitNet)                                                                               |
| twuii   | int     | RestartDialog(_In_opt_ HWND hParent, _In_opt_ PCWSTR pszPrompt, DWORD dwFlags)                                                            |
| twuiiii | int     | RestartDialogEx(_In_opt_ HWND hParent, _In_opt_ PCWSTR pszPrompt, DWORD dwFlags, DWORD dwReasonCode)                                      |
| wi      | HRESULT | SetCurrentProcessExplicitAppUserModelID(_In_ PCWSTR AppID)                                                                                |
| wti     | HRESULT | SHAddDefaultPropertiesByExt(_In_ PCWSTR pszExt, _In_ IPropertyStore *pPropStore)                                                          |
| ttui    | UINT    | SHAddFromPropSheetExtArray(_In_ HPSXA hpsxa, _In_ LPFNADDPROPSHEETPAGE lpfAddPage, LPARAM lParam)                                         |
| uiti    | VOID    | SHAddToRecentDoc(UINT uFlags, _In_opt_ LPCVOID pv)                                                                                        |
| tt      | LPVOID  | SHAlloc(_In_ SIZE_T cb)                                                                                                                   |

|           |                  |                                                                                                                                                                                               |
|-----------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uitt      | UINT_PTR         | SHAppBarMessage(_In_ DWORD dwMessage, _Inout_ PAPPBARDATA pData)                                                                                                                              |
| witi      | HRESULT          | SHAssocEnumHandlers(_In_ PCWSTR pszExtra, _In_ ASSOC_FILTER afFilter, _Out_ IEnumAssocHandlers **ppEnumHandler)                                                                               |
| wtti      | HRESULT          | SHAssocEnumHandlersForProtocolByApplication(_In_ PCWSTR protocol, _In_ REFIID riid, _Out_ void **enumHandlers)                                                                                |
| tttti     | HRESULT          | SHBindToFolderIDLListParent(_In_opt_ IShellFolder *psfRoot, _In_ PCUIDLIST_RELATIVE pidl, _In_ REFIID riid, _Out_ void **ppv, _Out_opt_ PCITEMID_CHILD *ppidlLast)                            |
| ttttti    | HRESULT          | SHBindToFolderIDLListParentEx(_In_opt_ IShellFolder *psfRoot, _In_ PCUIDLIST_RELATIVE pidl, _In_opt_ IBindCtx *ppbc, _In_ REFIID riid, _Out_ void **ppv, _Out_opt_ PCITEMID_CHILD *ppidlLast) |
| ttttti    | HRESULT          | SHBindToObject(IShellFolder *psf, PCUIDLIST_RELATIVE pidl, _In_ IBindCtx *pbc, REFIID riid, _Out_ void **ppv)                                                                                 |
| tttti     | HRESULT          | SHBindToParent(_In_ PCIDLIST_ABSOLUTE pidl, _In_ REFIID riid, _Out_ VOID **ppv, _Out_ PCITEMID_CHILD *ppidlLast)                                                                              |
| tt        | PIDLIST_ABSOLUTE | SHBrowseForFolder(_In_ LPBROWSEINFO lpbi)                                                                                                                                                     |
| tt        | PIDLIST_ABSOLUTE | SHBrowseForFolderA(_In_ LPBROWSEINFO lpbi)                                                                                                                                                    |
| tt        | PIDLIST_ABSOLUTE | SHBrowseForFolderW(_In_ LPBROWSEINFO lpbi)                                                                                                                                                    |
| tuittt    | HANDLE           | SHChangeNotification_Lock(_In_ HANDLE hChange, DWORD dwProcId, _Out_opt_ PIDLIST_ABSOLUTE **pppidl, _Out_opt_ LONG *plEvent)                                                                  |
| ti        | BOOL             | SHChangeNotification_Unlock(_In_ HANDLE hLock)                                                                                                                                                |
| uiuitti   | VOID             | SHChangeNotify(LONG wEventId, UINT uFlags, _In_opt_ LPCVOID dwItem1, _In_opt_ LPCVOID dwItem2)                                                                                                |
| uii       | BOOL             | SHChangeNotifyDeregister(ULONG ulID)                                                                                                                                                          |
| tiuiiitui | ULONG            | SHChangeNotifyRegister(_In_ HWND hwnd, int fSources, LONG fEvents, UINT wParam, int cEntries, _In_ const SHChangeNotifyEntry *pshcne)                                                         |
| uii       | VOID             | SHChangeNotifyRegisterThread(SCNRT_STATUS status)                                                                                                                                             |
| tiit      | PIDLIST_ABSOLUTE | SHCloneSpecialIDLList(HWND hwndOwner, _In_ int csidl, _In_ BOOL fCreate)                                                                                                                      |
| wti       | HRESULT          | SHCLSIDFromString(_In_ PCWSTR psz, _Out_ CLSID *pClsid)                                                                                                                                       |
| wtttti    | HRESULT          | SHCoCreateInstance(_In_opt_ PCWSTR pszCLSID, _In_opt_ const CLSID *pclsid, _In_opt_ IUnknown *pUnkOuter, _In_ REFIID riid, _Out_ void **ppv)                                                  |
|           |                  | SHCreateAssociationRegistration(_In_ REFIID riid, _Out_                                                                                                                                       |

|          |         |                                                                                                                                                                                     |
|----------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii      | HRESULT | void **ppv)                                                                                                                                                                         |
| tuittiti | HRESULT | SHCreateDataObject(_In_opt_ PCIDLIST_ABSOLUTE pidlFolder, _In_ UINT cidl, _In_opt_ PCUITEMID_CHILD_ARRAY apidl, _In_opt_ IDataObject *pdtInner, _In_ REFIID riid, _Out_ void **ppv) |
| titi     | HRESULT | SHCreateDefaultContextMenu(_In_ const DEFCONTEXTMENU *pdcM, REFIID riid, _Out_ void **ppv)                                                                                          |
| tii      | HRESULT | SHCreatesDefaultExtractIcon(REFIID riid, _Out_ void **ppv)                                                                                                                          |
| tii      | HRESULT | SHCreateDefaultPropertiesOp(_In_ IShellItem *psi, _Out_ IFileOperation **ppFileOp)                                                                                                  |
| twi      | int     | SHCreateDirectory(_In_opt_ HWND hwnd, _In_ PCWSTR pszPath)                                                                                                                          |
| tsti     | int     | SHCreateDirectoryEx(_In_opt_ HWND hwnd, _In_ LPCTSTR pszPath, _In_opt_ const SECURITY_ATTRIBUTES *psa)                                                                              |
| tati     | int     | SHCreateDirectoryExA(_In_opt_ HWND hwnd, _In_ LPCSTR pszPath, _In_opt_ const SECURITY_ATTRIBUTES *psa)                                                                              |
| twti     | int     | SHCreateDirectoryExW(_In_opt_ HWND hwnd, _In_ LPCWSTR pszPath, _In_opt_ const SECURITY_ATTRIBUTES *psa)                                                                             |
| suitti   | HRESULT | SHCreatesFileExtractIcon(_In_ LPCTSTR pszFile, _In_ DWORD dwFileAttributes, _In_ REFIID riid, void **ppv)                                                                           |
| wuitti   | HRESULT | SHCreateFileExtractIconW(_In_ LPCWSTR pszFile, _In_ DWORD dwFileAttributes, _In_ REFIID riid, void **ppv)                                                                           |
| titi     | HRESULT | SHCreateItemFromDLis(_In_ PCIDLIST_ABSOLUTE pidl, _In_ REFIID riid, _Out_ void **ppv)                                                                                               |
| wtiti    | HRESULT | SHCreateItemFromParsingName(_In_ PCWSTR pszPath, _In_ IBindCtx *pbc, _In_ REFIID riid, _Out_ void **ppv)                                                                            |
| twtiti   | HRESULT | SHCreateItemFromRelativeName(_In_ IShellItem *psiParent, _In_ PCWSTR pszName, _In_ IBindCtx *pbc, _In_ REFIID riid, _Out_ void **ppv)                                               |
| tuiwti   | HRESULT | SHCreateItemInKnownFolder(_In_ REFKNOWNFOLDERID kfid, DWORD dwKFFlags, _In_opt_ PCWSTR pszItem, _In_ REFIID riid, _Out_ void **ppv)                                                 |
| tttiti   | HRESULT | SHCreatesItemWithParent(_In_ PCIDLIST_ABSOLUTE pidlParent, _In_ IShellFolder *psfParent, _In_ PCUITEMID_CHILD pidl, _In_ REFIID riid, _Out_ void **ppvItem)                         |
| ti       | BOOL    | SHCreateProcessAsUserW(_Inout_ PSHCREATEPROCESSINFO pscpi)                                                                                                                          |
| twuit    | HPSXA   | SHCreatePropSheetExtArray(_In_ HKEY hkey, _In_opt_ PCWSTR pszSubkey, UINT max_iface)                                                                                                |
|          |         |                                                                                                                                                                                     |

|           |         |                                                                                                                                                                            |
|-----------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti        | HRESULT | SHCreateQueryCancelAutoPlayMoniker(_Out_ IMoniker **ppmoniker)                                                                                                             |
| tti       | HRESULT | SHCreateShellFolderView(_In_ const SFV_CREATE *pcsfv, _Out_ IShellView **ppsv)                                                                                             |
| tti       | HRESULT | SHCreateShellFolderViewEx(_In_ CSFV *pcsfv, _Out_ IShellView **ppsv)                                                                                                       |
| ttti      | HRESULT | SHCreateShellItem(_In_opt_ PCIDLIST_ABSOLUTE pidlParent, _In_opt_ IShellFolder *psfParent, _In_ PCUITEMID_CHILD pidl, _Out_ IShellItem **ppsi)                             |
| ttuitti   | HRESULT | SHCreateShellItemArray(_In_ PCIDLIST_ABSOLUTE pidlParent, _In_ IShellFolder *psf, _In_ UINT cidl, _In_ PCUITEMID_CHILD_ARRAY ppidl, _Out_ IShellItemArray **ppsiItemArray) |
| ttti      | HRESULT | SHCreateShellItemArrayFromDataObject(_In_ IDataObject *pdo, _In_ REFIID riid, _Out_ void **ppv)                                                                            |
| uitti     | HRESULT | SHCreateShellItemArrayFromDLLists(_In_ UINT cidl, _In_ PCIDLIST_ABSOLUTE_ARRAY rgpidl, _Out_ IShellItemArray **ppsiItemArray)                                              |
| ttti      | HRESULT | SHCreateShellItemArrayFromShellItem(_In_ IShellItem *psi, _In_ REFIID riid, _Out_ void **ppv)                                                                              |
| uiuiti    | HRESULT | SHCreateSidEnumFmtEtc(_In_ UINT cfmt, const FORMATETC afmt[], _Out_ IEnumFORMATETC **ppenumFormatEtc)                                                                      |
| siuittuui | HRESULT | SHDefExtractIcon(_In_ LPCTSTR pszIconFile, int iIndex, _In_ UINT uFlags, _Out_opt_ HICON *phiconLarge, _Out_opt_ HICON *phiconSmall, UINT nIconSize)                       |
| aiuittuui | HRESULT | SHDefExtractIconA(_In_ LPCSTR pszIconFile, int iIndex, _In_ UINT uFlags, _Out_opt_ HICON *phiconLarge, _Out_opt_ HICON *phiconSmall, UINT nIconSize)                       |
| wiuittuui | HRESULT | SHDefExtractIconW(_In_ LPCWSTR pszIconFile, int iIndex, _In_ UINT uFlags, _Out_opt_ HICON *phiconLarge, _Out_opt_ HICON *phiconSmall, UINT nIconSize)                      |
| ti        | VOID    | SHDestroyPropSheetExtArray(_In_ HPSXA hpsxa)                                                                                                                               |
| tttuiti   | HRESULT | SHDoDragDrop(_In_ HWND hwnd, _In_ IDataObject *pdtobj, _In_ IDropSource *pdsr, _In_ DWORD dwEffect, _Out_ DWORD *pdwEffect)                                                |
| wiuui     | int     | Shell_GetCachedImageIndex(_In_ PCWSTR pwszIconPath, int iIconIndex, UINT uIconFlags)                                                                                       |
| aiuui     | int     | Shell_GetCachedImageIndexA(_In_ PCWSTR pwszIconPath, int iIconIndex, UINT uIconFlags)                                                                                      |
| wiuui     | int     | Shell_GetCachedImageIndexW(_In_ PCWSTR pwszIconPath, int iIconIndex, UINT uIconFlags)                                                                                      |
| tti       | BOOL    | Shell_GetImageLists(_In_ HIMAGELIST *phiml, _In_                                                                                                                           |

|              |           |                                                                                                                                                                      |
|--------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |           | HIMAGELIST *phimlSmall)                                                                                                                                              |
| ttuiuiuiuiui | UINT      | Shell_MergeMenus(_In_ HMENU hmDst, _In_ HMENU hmSrc, UINT uInsert, UINT uIDAdjust, UINT uIDAdjustMax, ULONG uFlags)                                                  |
| uiti         | BOOL      | Shell_NotifyIcon(_In_ DWORD dwMessage, _In_ NOTIFYICONDATA lpdata)                                                                                                   |
| uiti         | BOOL      | Shell_NotifyIconA(_In_ DWORD dwMessage, _In_ NOTIFYICONDATA lpdata)                                                                                                  |
| titi         | HRESULT   | Shell_NotifyIconGetRect(_In_ const NOTIFYICONIDENTIFIER *identifier, _Out_ RECT *iconLocation)                                                                       |
| uiti         | BOOL      | Shell_NotifyIconW(_In_ DWORD dwMessage, _In_ NOTIFYICONDATA lpdata)                                                                                                  |
| tssti        | int       | ShellAbout(_In_opt_ HWND hWnd, _In_ LPCTSTR szApp, _In_opt_ LPCTSTR szOtherStuff, _In_opt_ HICON hIcon)                                                              |
| taati        | int       | ShellAboutA(_In_opt_ HWND hWnd, _In_ LPCSTR szApp, _In_opt_ LPCSTR szOtherStuff, _In_opt_ HICON hIcon)                                                               |
| twwti        | int       | ShellAboutW(_In_opt_ HWND hWnd, _In_ LPCWSTR szApp, _In_opt_ LPCWSTR szOtherStuff, _In_opt_ HICON hIcon)                                                             |
| tssssit      | HINSTANCE | ShellExecute(_In_opt_ HWND hwnd, _In_opt_ LPCTSTR lpOperation, _In_ LPCTSTR lpFile, _In_opt_ LPCTSTR lpParameters, _In_opt_ LPCTSTR lpDirectory, _In_ INT nShowCmd)  |
| taaaait      | HINSTANCE | ShellExecuteA(_In_opt_ HWND hwnd, _In_opt_ LPCSTR lpOperation, _In_ LPCSTR lpFile, _In_opt_ LPCSTR lpParameters, _In_opt_ LPCSTR lpDirectory, _In_ INT nShowCmd)     |
| ti           | BOOL      | ShellExecuteEx(_Inout_ SHELLEXECUTEINFO *pExecInfo)                                                                                                                  |
| ti           | BOOL      | ShellExecuteExA(_Inout_ SHELLEXECUTEINFO *pExecInfo)                                                                                                                 |
| ti           | BOOL      | ShellExecuteExW(_Inout_ SHELLEXECUTEINFO *pExecInfo)                                                                                                                 |
| twwwwit      | HINSTANCE | ShellExecuteW(_In_opt_ HWND hwnd, _In_opt_ LPCWSTR lpOperation, _In_ LPCWSTR lpFile, _In_opt_ LPCWSTR lpParameters, _In_opt_ LPCWSTR lpDirectory, _In_ INT nShowCmd) |
| tsuii        | HRESULT   | SHEmptyRecycleBin(_In_opt_ HWND hwnd, _In_opt_ LPCTSTR pszRootPath, DWORD dwFlags)                                                                                   |
| tauii        | HRESULT   | SHEmptyRecycleBinA(_In_opt_ HWND hwnd, _In_opt_ LPCSTR pszRootPath, DWORD dwFlags)                                                                                   |
| twuii        | HRESULT   | SHEmptyRecycleBinW(_In_opt_ HWND hwnd, _In_opt_ LPCWSTR pszRootPath, DWORD dwFlags)                                                                                  |

|              |               |                                                                                                                                                                               |
|--------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuisii       | HRESULT       | SHEnumerateUnreadMailAccounts(_In_opt_ HKEY hKeyUser, DWORD dwIndex, _Out_ LPTSTR pszMailAddress, int cchMailAddress)                                                         |
| tuiwii       | HRESULT       | SHEnumerateUnreadMailAccountsW(_In_opt_ HKEY hKeyUser, DWORD dwIndex, _Out_ LPWSTR pszMailAddress, int cchMailAddress)                                                        |
| wwwwi        | HRESULT       | SHEvaluateSystemCommandTemplate(_In_ PCWSTR pszCmdTemplate, _Out_ PWSTR *ppszApplication, _Out_opt_ PWSTR *ppszCommandLine, _Out_opt_ PWSTR *ppszParameters)                  |
| wiiittuiuiui | UINT          | SHExtractIconsW(_In_ LPCWSTR pszFileName, _In_ int nIconIndex, _In_ int cxIcon, _In_ int cyIcon, _Out_ HICON *phIcon, _Out_ UINT *pIconId, _In_ UINT nIcons, _In_ UINT flags) |
| ti           | int           | SHFileOperation(_Inout_ LPSHFILEOPSTRUCT lpFileOp)                                                                                                                            |
| ti           | int           | SHFileOperationA(_Inout_ LPSHFILEOPSTRUCT lpFileOp)                                                                                                                           |
| ti           | int           | SHFileOperationW(_Inout_ LPSHFILEOPSTRUCT lpFileOp)                                                                                                                           |
| ttuiuit      | IContextMenu* | SHFind_InitMenuPopup(_In_ HMENU hmenu, _In_opt_ HWND hwnd, UINT idCmdFirst, UINT idCmdLast)                                                                                   |
| tii          | BOOL          | SHFindFile(_In_opt_ PCIDLIST_ABSOLUTE pidlFolder, _In_opt_ PCIDLIST_ABSOLUTE pidlSaveFile)                                                                                    |
| i            | VOID          | SHFlushSFCache(void)                                                                                                                                                          |
| tuiuiuiui    | DWORD         | SHFormatDrive(_In_ HWND hwnd, UINT drive, UINT fmtID, UINT options)                                                                                                           |
| ti           | VOID          | SHFree(_In_ VOID *pv)                                                                                                                                                         |
| ti           | VOID          | SHFreeNameMappings(_In_opt_ HANDLE hNameMappings)                                                                                                                             |
| tuitti       | HRESULT       | SHGetAttributesFromDataObject(_In_opt_ IDataObject *pdo, DWORD dwAttributeMask, _Out_opt_ DWORD *pdwAttributes, _Out_opt_ UINT *pcItems)                                      |
| ttitii       | HRESULT       | SHGetDataFromIDLis(_In_ IShellFolder *psf, _In_ PCUITEMID_CHILD pidl, int nFormat, _Out_ void *pv, int cb)                                                                    |
| ttitii       | HRESULT       | SHGetDataFromIDLisA(_In_ IShellFolder *psf, _In_ PCUITEMID_CHILD pidl, int nFormat, _Out_ void *pv, int cb)                                                                   |
| ttitii       | HRESULT       | SHGetDataFromIDLisW(_In_ IShellFolder *psf, _In_ PCUITEMID_CHILD pidl, int nFormat, _Out_ void *pv, int cb)                                                                   |
| ti           | HRESULT       | SHGetDesktopFolder(_Out_ IShellFolder **ppshf)                                                                                                                                |
| sttti        | BOOL          | SHGetDiskFreeSpace(LPCTSTR pszVolume, ULARGE_INTEGER *pqwFreeCaller, ULARGE_INTEGER *pqwTot, ULARGE_INTEGER *pqwFree)                                                         |
|              |               |                                                                                                                                                                               |

|           |           |                                                                                                                                                                                                                     |
|-----------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| attti     | BOOL      | SHGetDiskFreeSpaceA(LPCSTR pszVolume, ULARGE_INTEGER *pqwFreeCaller, ULARGE_INTEGER *pqwTot, ULARGE_INTEGER *pqwFree)                                                                                               |
| sttti     | BOOL      | SHGetDiskFreeSpaceEx(_In_ LPCTSTR pszDirectoryName, _Out_opt_ ULARGE_INTEGER *pulFreeBytesAvailableToCaller, _Out_opt_ ULARGE_INTEGER *pulTotalNumberOfBytes, _Out_opt_ ULARGE_INTEGER *pulTotalNumberOfFreeBytes)  |
| attti     | BOOL      | SHGetDiskFreeSpaceExA(_In_ LPCSTR pszDirectoryName, _Out_opt_ ULARGE_INTEGER *pulFreeBytesAvailableToCaller, _Out_opt_ ULARGE_INTEGER *pulTotalNumberOfBytes, _Out_opt_ ULARGE_INTEGER *pulTotalNumberOfFreeBytes)  |
| wttti     | BOOL      | SHGetDiskFreeSpaceExW(_In_ LPCWSTR pszDirectoryName, _Out_opt_ ULARGE_INTEGER *pulFreeBytesAvailableToCaller, _Out_opt_ ULARGE_INTEGER *pulTotalNumberOfBytes, _Out_opt_ ULARGE_INTEGER *pulTotalNumberOfFreeBytes) |
| wti       | HRESULT   | SHGetDriveMedia(_In_ PCWSTR pszDrive, _Out_ DWORD *pdwMediaContent)                                                                                                                                                 |
| suituiuit | DWORD_PTR | SHGetFileInfo(_In_ LPCTSTR pszPath, DWORD dwFileAttributes, _Inout_ SHFILEINFO *psfi, UINT cbFileInfo, UINT uFlags)                                                                                                 |
| autiuiuit | DWORD_PTR | SHGetFileInfoA(_In_ LPCSTR pszPath, DWORD dwFileAttributes, _Inout_ SHFILEINFO *psfi, UINT cbFileInfo, UINT uFlags)                                                                                                 |
| wuituiuit | DWORD_PTR | SHGetFileInfoW(_In_ LPCWSTR pszPath, DWORD dwFileAttributes, _Inout_ SHFILEINFO *psfi, UINT cbFileInfo, UINT uFlags)                                                                                                |
| tituiti   | HRESULT   | SHGetFolderLocation(_In_ HWND hwndOwner, _In_ int nFolder, _In_ HANDLE hToken, _Reserved_ DWORD dwReserved, _Out_ PIDLIST_ABSOLUTE *ppidl)                                                                          |
| tituisi   | HRESULT   | SHGetFolderPath(_In_ HWND hwndOwner, _In_ int nFolder, _In_ HANDLE hToken, _In_ DWORD dwFlags, _Out_ LPTSTR pszPath)                                                                                                |
| tituiai   | HRESULT   | SHGetFolderPathA(_In_ HWND hwndOwner, _In_ int nFolder, _In_ HANDLE hToken, _In_ DWORD dwFlags, _Out_ LPSTR pszPath)                                                                                                |
| tituissi  | HRESULT   | SHGetFolderPathAndSubDir(_In_ HWND hwnd, _In_ int csidl, _In_ HANDLE hToken, _In_ DWORD dwFlags, _In_ LPCTSTR pszSubDir, _Out_ LPTSTR pszPath)                                                                      |
| tituiaai  | HRESULT   | SHGetFolderPathAndSubDirA(_In_ HWND hwnd, _In_ int csidl, _In_ HANDLE hToken, _In_ DWORD dwFlags, _In_ LPCSTR pszSubDir, _Out_ LPSTR pszPath)                                                                       |
| tituiwwi  | HRESULT   | SHGetFolderPathAndSubDirW(_In_ HWND hwnd, _In_ int csidl, _In_ HANDLE hToken, _In_ DWORD dwFlags, _In_                                                                                                              |

|         |         |                                                                                                                                          |
|---------|---------|------------------------------------------------------------------------------------------------------------------------------------------|
|         |         | LPCWSTR pszSubDir, _Out_ LPWSTR pszPath)                                                                                                 |
| tituiwi | HRESULT | SHGetFolderPathW(_In_ HWND hwndOwner, _In_ int nFolder, _In_ HANDLE hToken, _In_ DWORD dwFlags, _Out_ LPWSTR pszPath)                    |
| sii     | int     | SHGetIconOverlayIndex(_In_opt_ LPCTSTR pszIconPath, int iIconIndex)                                                                      |
| aii     | int     | SHGetIconOverlayIndexA(_In_opt_ LPCSTR pszIconPath, int iIconIndex)                                                                      |
| wii     | int     | SHGetIconOverlayIndexW(_In_opt_ LPCWSTR pszIconPath, int iIconIndex)                                                                     |
| tii     | HRESULT | SHGetIDListFromObject(_In_ IUnknown *punk, _Out_ PIDLIST_ABSOLUTE *ppidl)                                                                |
| itii    | HRESULT | SHGetImageList(_In_ int iImageList, _In_ REFIID riid, _Out_ void **ppv)                                                                  |
| ti      | HRESULT | SHGetInstanceExplorer(_Out_ IUnknown **ppunk)                                                                                            |
| tuitii  | HRESULT | SHGetItemFromDataObject(_In_ IDataObject *pdtobj, _In_ DATAOBJ_GET_ITEM_FLAGS dwFlags, _In_ REFIID riid, _Out_ void **ppv)               |
| tiii    | HRESULT | SHGetItemFromObject(_In_ IUnknown *punk, _In_ REFIID riid, _Out_ void **ppv)                                                             |
| tuitii  | HRESULT | SHGetKnownFolderIDList(_In_ REFKNOWNFOLDERID rfid, _In_ DWORD dwFlags, _In_ HANDLE hToken, _Out_ PIDLIST_ABSOLUTE *ppidl)                |
| tuiiii  | HRESULT | SHGetKnownFolderItem(_In_ REFKNOWNFOLDERID rfid, _In_ KNOWN_FOLDER_FLAG dwFlags, _In_ HANDLE hToken, _In_ REFIID riid, _Out_ void **ppv) |
| tuiwi   | HRESULT | SHGetKnownFolderPath(_In_ REFKNOWNFOLDERID rfid, _In_ DWORD dwFlags, _In_opt_ HANDLE hToken, _Out_ PWSTR *ppszPath)                      |
| wtuiti  | HRESULT | SHGetLocalizedName(_In_ PCWSTR pszPath, _Out_ PWSTR pszResModule, UINT cch, _Out_ int *pidsRes)                                          |
| ti      | HRESULT | SHGetMalloc(LPALLOC *ppMalloc)                                                                                                           |
| tuiwi   | HRESULT | SHGetNameFromIDList(_In_ PCIDLIST_ABSOLUTE pidl, _In_ SIGDN sigdnName, _Out_ PWSTR *ppszName)                                            |
| ssstuii | BOOL    | SHGetNewLinkInfo(_In_ LPCTSTR pszLinkTo, _In_ LPCTSTR pszDir, _Out_ LPTSTR pszName, _Out_ BOOL *pfMustCopy, _In_ UINT uFlags)            |
| aaatuii | BOOL    | SHGetNewLinkInfoA(_In_ LPCSTR pszLinkTo, _In_ LPCSTR pszDir, _Out_ LPSTR pszName, _Out_ BOOL *pfMustCopy, _In_ UINT uFlags)              |
| wwwtuii | BOOL    | SHGetNewLinkInfoW(_In_ LPCWSTR pszLinkTo, _In_ LPCWSTR pszDir, _Out_ LPWSTR pszName, _Out_ BOOL *pfMustCopy, _In_ UINT uFlags)           |

|         |         |                                                                                                                                                      |
|---------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsi     | BOOL    | SHGetPathFromIDList(_In_ PCIDLIST_ABSOLUTE pidl, _Out_ LPTSTR pszPath)                                                                               |
| tai     | BOOL    | SHGetPathFromIDListA(_In_ PCIDLIST_ABSOLUTE pidl, _Out_ LPSTR pszPath)                                                                               |
| ttuiii  | BOOL    | SHGetPathFromIDListEx(_In_ PCIDLIST_ABSOLUTE pidl, _Out_ PWSTR pszPath, DWORD cchPath, GPFIDL_FLAGS uOpts)                                           |
| twi     | BOOL    | SHGetPathFromIDListW(_In_ PCIDLIST_ABSOLUTE pidl, _Out_ LPWSTR pszPath)                                                                              |
| ttti    | HRESULT | SHGetPropertyStoreForWindow(_In_ HWND hwnd, _In_ REFIID riid, _Out_ void **ppv)                                                                      |
| tuitti  | HRESULT | SHGetPropertyStoreFromIDList(_In_ PCIDLIST_ABSOLUTE pidl, _In_ GETPROPERTYSTOREFLAGS flags, _In_ REFIID riid, _Out_ void **ppv)                      |
| wtuitti | HRESULT | SHGetPropertyStoreFromParsingName(_In_ PCWSTR pszPath, _In_opt_ IBindCtx *pbc, _In_ GETPROPERTYSTOREFLAGS flags, _In_ REFIID riid, _Out_ void **ppv) |
| ttti    | HRESULT | SHGetRealIDL(_In_ IShellFolder *psf, _In_ PCITEMID_CHILD pidlSimple, _Out_ PITEMID_CHILD *ppidlReal)                                                 |
| tsuii   | HRESULT | SHGetSetFolderCustomSettings(_Inout_ LPSHFOLDERCUSTOMSETTINGS pfc, _In_ PCTSTR pszPath, DWORD dwReadWrite)                                           |
| tuiii   | VOID    | SHGetSetSettings(_Inout_ LPSHELLSTATE lps, _In_ DWORD dwMask, _In_ BOOL bSet)                                                                        |
| tuii    | VOID    | SHGetSettings(LPSHELLFLAGSTATE lpsfs, DWORD dwMask)                                                                                                  |
| titi    | HRESULT | SHGetSpecialFolderLocation(_In_ HWND hwndOwner, _In_ int nFolder, _Out_ PIDLIST_ABSOLUTE *ppidl)                                                     |
| tsiii   | BOOL    | SHGetSpecialFolderPath(HWND hwndOwner, _Out_ LPTSTR lpszPath, _In_ int csidl, _In_ BOOL fCreate)                                                     |
| taiii   | BOOL    | SHGetSpecialFolderPathA(HWND hwndOwner, _Out_ LPSTR lpszPath, _In_ int csidl, _In_ BOOL fCreate)                                                     |
| twiii   | BOOL    | SHGetSpecialFolderPathW(HWND hwndOwner, _Out_ LPWSTR lpszPath, _In_ int csidl, _In_ BOOL fCreate)                                                    |
| uiuiti  | HRESULT | SHGetStockIconInfo(SHSTOCKICONID siid, UINT uFlags, _Inout_ SHSTOCKICONINFO *psii)                                                                   |
| ttti    | HRESULT | SHGetTemporaryPropertyForItem(_In_ IShellItem *psi, REFPROPERTYKEY pk, _Out_ PROPVARIANT *ppropvarInk)                                               |
|         |         | SHGetUnreadMailCount(_In_opt_ HKEY hKeyUser, _In_opt_ LPCTSTR pszMailAddress, _Out_opt_ DWORD *pdwCount,                                             |

|          |         |                                                                                                                                                                                                                |
|----------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsstsi   | HRESULT | _Out_opt_ FILETIME *pFileTime, _Out_opt_ LPCTSTR pszShellExecuteCommand, int cchShellExecuteCommand)                                                                                                           |
| twttwii  | HRESULT | SHGetUnreadMailCountW(_In_opt_ HKEY hKeyUser, _In_opt_ LPCWSTR pszMailAddress, _Out_opt_ DWORD *pdwCount, _Out_opt_ FILETIME *pFileTime, _Out_opt_ LPCWSTR pszShellExecuteCommand, int cchShellExecuteCommand) |
| ti       | int     | SHHandleUpdateImage(_In_ PCIDLIST_ABSOLUTE pidlExtra)                                                                                                                                                          |
| wtti     | HRESULT | SHILCreateFromPath(_In_ PCWSTR pszPath, _Out_ PIDLIST_ABSOLUTE *ppidl, _Inout_opt_ DWORD *rgflnOut)                                                                                                            |
| tuiisii  | BOOL    | SHInvokePrinterCommand(_In_opt_ HWND hwnd, UINT uAction, _In_ LPCTSTR lpBuf1, _In_opt_ LPCTSTR lpBuf2, BOOL fModal)                                                                                            |
| tuiiaai  | BOOL    | SHInvokePrinterCommandA(_In_opt_ HWND hwnd, UINT uAction, _In_ LPCSTR lpBuf1, _In_opt_ LPCSTR lpBuf2, BOOL fModal)                                                                                             |
| tuiwwii  | BOOL    | SHInvokePrinterCommandW(_In_opt_ HWND hwnd, UINT uAction, _In_ LPCWSTR lpBuf1, _In_opt_ LPCWSTR lpBuf2, BOOL fModal)                                                                                           |
| wti      | HRESULT | SHIsFileAvailableOffline(_In_ PCWSTR pszPath, _Out_opt_ LPDWORD pdwStatus)                                                                                                                                     |
| tii      | HRESULT | SHLimitInputEdit(_In_opt_ HWND hwndEdit, _In_ IShellFolder *psf)                                                                                                                                               |
| ti       | HRESULT | SHLoadInProc(_In_ REFCLSID rclsid)                                                                                                                                                                             |
| i        | HRESULT | SHLoadNonloadedIconOverlayIdentifiers(void)                                                                                                                                                                    |
| titi     | int     | SHMapPIDLToSystemImageListIndex(_In_ IShellFolder *psf, _In_ PCUITEMID_CHILD pidl, _Out_opt_ int *piIndex)                                                                                                     |
| tuii     | HRESULT | SHMultiFileProperties(_In_ IDataObject *pdtobj, DWORD dwFlags)                                                                                                                                                 |
| tuiwwi   | BOOL    | SHObjectProperties(_In_ HWND hwnd, _In_ DWORD shopObjectType, _In_ PCWSTR pszObjectName, _In_ PCWSTR pszPropertyPage)                                                                                          |
| tuitui   | HRESULT | SHOpenFolderAndSelectItems(_In_ PCIDLIST_ABSOLUTE pidlFolder, UINT cidl, _In_opt_ PCUITEMID_CHILD_ARRAY apidl, DWORD dwFlags)                                                                                  |
| stuittsi | BOOL    | SHOpenPropSheet(_In_opt_ LPCTSTR pszCaption, _In_opt_ HKEY ahkeys[], UINT ckeys, _In_opt_ const CLSID *pclsidDef, _In_ IDataObject *pdtobj, _In_opt_ IShellBrowser *psb, _In_opt_ LPCTSTR pStartPage)          |
| wtuittwi | BOOL    | SHOpenPropSheetW(_In_opt_ LPCWSTR pszCaption, _In_opt_ HKEY ahkeys[], UINT ckeys, _In_opt_ const CLSID                                                                                                         |

|              |         |                                                                                                                                                                                                                     |
|--------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |         | *pclsidDef, _In_ IDataObject *pdtobj, _In_opt_ IShellBrowser *psb, _In_opt_ LPCWSTR pStartPage)                                                                                                                     |
| tti          | HRESULT | SHOpenWithDialog(_In_opt_ HWND hwndParent, _In_ const OPENASINFO *poainfo)                                                                                                                                          |
| wttuiti      | HRESULT | SHParseDisplayName(_In_ LPCWSTR pszName, _In_opt_ IBindCtx *pbc, _Out_ PIDLIST_ABSOLUTE *ppidl, _In_ SFGAOF sfgaoIn, _Out_opt_ SFGAOF *psfgaoOut)                                                                   |
| ttuii        | HRESULT | SHPathPrepareForWrite(_In_opt_ HWND hwnd, _In_opt_ IUnknown *punkEnableModless, _In_ LPCTSTR pszPath, DWORD dwFlags = SHPPFW_DEFAULT)                                                                               |
| ttauui       | HRESULT | SHPathPrepareForWriteA(_In_opt_ HWND hwnd, _In_opt_ IUnknown *punkEnableModless, _In_ LPCSTR pszPath, DWORD dwFlags = SHPPFW_DEFAULT)                                                                               |
| ttwuii       | HRESULT | SHPathPrepareForWriteW(_In_opt_ HWND hwnd, _In_opt_ IUnknown *punkEnableModless, _In_ LPCWSTR pszPath, DWORD dwFlags = SHPPFW_DEFAULT)                                                                              |
| tttuiuuiitti | HRESULT | SHPropStgCreate(_In_ IPropertySetStorage *psstg, _In_ REFFMTID fmtid, _In_opt_ const CLSID *pclsid, DWORD grfFlags, DWORD grfMode, DWORD dwDisposition, _Out_ IPropertyStorage **ppstg, _Out_opt_ UINT *puCodePage) |
| tuiuitti     | HRESULT | SHPropStgReadMultiple(_In_ IPropertyStorage *pps, UINT uCodePage, ULONG cpspec, _In_ PROPSPEC const rgpspec[], _Out_ PROPVARIANT rgvar[])                                                                           |
| ttuittui     | HRESULT | SHPropStgWriteMultiple(_In_ IPropertyStorage *pps, _Inout_opt_ UINT *uCodePage, ULONG cpspec, _In_ PROPSPEC const rgpspec[], _Inout_ PROPVARIANT rgvar[], PROPID propidNameFirst)                                   |
| sti          | HRESULT | SHQueryRecycleBin(_In_opt_ LPCTSTR pszRootPath, _Inout_ LPSHQUERYRBINFO pSHQueryRBInfo)                                                                                                                             |
| ati          | HRESULT | SHQueryRecycleBinA(_In_opt_ LPCSTR pszRootPath, _Inout_ LPSHQUERYRBINFO pSHQueryRBInfo)                                                                                                                             |
| wti          | HRESULT | SHQueryRecycleBinW(_In_opt_ LPCWSTR pszRootPath, _Inout_ LPSHQUERYRBINFO pSHQueryRBInfo)                                                                                                                            |
| ti           | HRESULT | SHQueryUserNotificationState(_Out_ QUERY_USER_NOTIFICATION_STATE *pquns)                                                                                                                                            |
| wi           | HRESULT | SHRemoveLocalizedName(_In_ PCWSTR pszPath)                                                                                                                                                                          |
| tuittui      | UINT    | SHReplaceFromPropSheetExtArray(_In_ HPSXA hpsxa, UINT uPageID, _In_ LPFNADDPROPSHEETPAGE lpfReplaceWith, LPARAM lParam)                                                                                             |
| ti           | HRESULT | SHResolveLibrary(_In_ IShellItem *psiLibrary)                                                                                                                                                                       |
| uiui         | DWORD   | SHRestricted(RESTRICTIONS rest)                                                                                                                                                                                     |
| ttuiti       | HRESULT | SHSetDefaultProperties(_In_opt_ HWND hwnd, _In_ IShellItem *psi, DWORD dwFileOpFlags, _In_opt_                                                                                                                      |

|         |                  |                                                                                                                                                                                     |
|---------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |                  | IFileOperationProgressSink *pfops)                                                                                                                                                  |
| ituisi  | HRESULT          | SHSetFolderPath(_In_ int csidl, _In_ HANDLE hToken, _Reserved_ DWORD dwFlags, _In_ LPCTSTR pszPath)                                                                                 |
| ituiai  | HRESULT          | SHSetFolderPathA(_In_ int csidl, _In_ HANDLE hToken, _Reserved_ DWORD dwFlags, _In_ LPCSTR pszPath)                                                                                 |
| ituiwi  | HRESULT          | SHSetFolderPathW(_In_ int csidl, _In_ HANDLE hToken, _Reserved_ DWORD dwFlags, _In_ LPCWSTR pszPath)                                                                                |
| ti      | VOID             | SHSetInstanceExplorer(_In_opt_ IUnknown *pUnk)                                                                                                                                      |
| tuitwi  | HRESULT          | SHSetKnownFolderPath(_In_ REFKNOWNFOLDERID rfid, _In_ DWORD dwFlags, _In_ HANDLE hToken, _In_ PCWSTR pszPath)                                                                       |
| wwii    | HRESULT          | SHSetLocalizedName(_In_ PCWSTR pszPath, _In_ PCWSTR pszResModule, int idsRes)                                                                                                       |
| ttti    | HRESULT          | SHSetTemporaryPropertyForItem(_In_ IShellItem *psi, _In_ REFPROPERTYKEY propkey, _In_ REFPROPVARIANT propvar)                                                                       |
| suisi   | HRESULT          | SHSetUnreadMailCount(_In_ LPCTSTR pszMailAddress, DWORD dwCount, _In_ LPCTSTR pszShellExecuteCommand)                                                                               |
| wuiwi   | HRESULT          | SHSetUnreadMailCountW(_In_ LPCWSTR pszMailAddress, DWORD dwCount, _In_ LPCWSTR pszShellExecuteCommand)                                                                              |
| tuitt   | LRESULT          | SHShellFolderView_Message(_In_ HWND hwndMain, UINT uMsg, LPARAM lParam)                                                                                                             |
| ttwwuii | HRESULT          | SHShowManageLibraryUI(_In_ IShellItem *psiLibrary, _In_opt_ HWND hwndOwner, _In_opt_ LPCWSTR pszTitle, _In_opt_ LPCWSTR pszInstruction, _In_ LIBRARYMANAGEDIALOGOPTIONS lmdOptions) |
| wt      | PIDLIST_ABSOLUTE | SHSimpleIDListFromPath(_In_ PCWSTR pszPath)                                                                                                                                         |
| tsuii   | HRESULT          | SHStartNetConnectionDialog(_In_opt_ HWND hwnd, _In_opt_ LPCTSTR pszRemoteName, DWORD dwType)                                                                                        |
| twuii   | HRESULT          | SHStartNetConnectionDialogW(_In_opt_ HWND hwnd, _In_opt_ LPCWSTR pszRemoteName, DWORD dwType)                                                                                       |
| tuii    | BOOL             | SHTestTokenMembership(_In_opt_ HANDLE hToken, ULONG ulRID)                                                                                                                          |
| siuiii  | VOID             | SHUpdateImage(_In_ LPCTSTR pszHashItem, _In_ int iIndex, _In_ UINT uFlags, _In_ int iImageIndex)                                                                                    |
| aiuiii  | VOID             | SHUpdateImageA(_In_ LPCSTR pszHashItem, _In_ int iIndex, _In_ UINT uFlags, _In_ int iImageIndex)                                                                                    |
| wiuiii  | VOID             | SHUpdateImageW(_In_ LPCWSTR pszHashItem, _In_ int iIndex, _In_ UINT uFlags, _In_ int iImageIndex)                                                                                   |
|         |                  |                                                                                                                                                                                     |

|         |           |                                                                                                                                                                             |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuii   | BOOL      | SHValidateUNC(_In_opt_ HWND hwndOwner, _Inout_ PWSTR pszFile, UINT fConnect)                                                                                                |
| ti      | BOOL      | SignalFileOpen(_In_ PCIDLIST_ABSOLUTE pidl)                                                                                                                                 |
| twuitti | HRESULT   | SigMakeUniqueName(_In_ IStorage *pstgParent, _In_ PCWSTR pszFileSpec, _In_ DWORD grfMode, _In_ REFIID riid, _Out_ void **ppv)                                               |
| wi      | BOOL      | Win32DeleteFile(_In_ PCWSTR pszFileName)                                                                                                                                    |
| tssstt  | HINSTANCE | WOWShellExecute(_In_ HWND hwnd, _In_ LPCTSTR lpOperation, _In_ LPCTSTR lpFile, _In_ LPCTSTR lpParameters, _In_ LPCTSTR lpDirectory, _In_ INT nShowCmd, void *lpfnCBWinExec) |
| ti      | BOOL      | WriteCabinetState(_In_ CABINETSTATE *pcs)                                                                                                                                   |

# Shlwapi.dll

|          |          |                                                                                                                                                                     |
|----------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tti      | HRESULT  | <i>AssocCreate</i> (_In_ CLSID clsid, _In_ REFIID riid, _Out_ void *ppv)                                                                                            |
| wttwi    | HRESULT  | <i>AssocGetPerceivedType</i> (_In_ PCWSTR pszExt, _Out_ PERCEIVED *ptype, _Out_ PERCEIVEDFLAG *pflag, _Out_opt_ PWSTR *ppszType)                                    |
| wi       | BOOL     | <i>AssocIsDangerous</i> (_In_ PCWSTR pszAssoc)                                                                                                                      |
| iissti   | HRESULT  | <i>AssocQueryKey</i> (_In_ ASSOCF flags, _In_ ASSOCKEY key, _In_ LPCTSTR pszAssoc, _In_ LPCTSTR pszExtra, _Out_ HKEY *phkeyOut)                                     |
| iiaati   | HRESULT  | <i>AssocQueryKeyA</i> (_In_ ASSOCF flags, _In_ ASSOCKEY key, _In_ LPCSTR pszAssoc, _In_ LPCSTR pszExtra, _Out_ HKEY *phkeyOut)                                      |
| iiwwti   | HRESULT  | <i>AssocQueryKeyW</i> (_In_ ASSOCF flags, _In_ ASSOCKEY key, _In_ LPCWSTR pszAssoc, _In_ LPCWSTR pszExtra, _Out_ HKEY *phkeyOut)                                    |
| iissti   | HRESULT  | <i>AssocQueryString</i> (_In_ ASSOCF flags, _In_ ASSOCSTR str, _In_ LPCTSTR pszAssoc, _In_opt_ LPCTSTR pszExtra, _Out_opt_ LPTSTR pszOut, _Inout_ DWORD *pcchOut)   |
| iiaaati  | HRESULT  | <i>AssocQueryStringA</i> (_In_ ASSOCF flags, _In_ ASSOCSTR str, _In_ LPCSTR pszAssoc, _In_opt_ LPCSTR pszExtra, _Out_opt_ LPSTR pszOut, _Inout_ DWORD *pcchOut)     |
| iitssti  | HRESULT  | <i>AssocQueryStringByKey</i> (_In_ ASSOCF flags, _In_ ASSOCSTR str, _In_ HKEY hkAssoc, _In_opt_ LPCTSTR pszExtra, _Out_opt_ LPTSTR pszOut, _Inout_ DWORD *pcchOut)  |
| iitaati  | HRESULT  | <i>AssocQueryStringByKeyA</i> (_In_ ASSOCF flags, _In_ ASSOCSTR str, _In_ HKEY hkAssoc, _In_opt_ LPCSTR pszExtra, _Out_opt_ LPSTR pszOut, _Inout_ DWORD *pcchOut)   |
| iitwwti  | HRESULT  | <i>AssocQueryStringByKeyW</i> (_In_ ASSOCF flags, _In_ ASSOCSTR str, _In_ HKEY hkAssoc, _In_opt_ LPCWSTR pszExtra, _Out_opt_ LPWSTR pszOut, _Inout_ DWORD *pcchOut) |
| iiwwwti  | HRESULT  | <i>AssocQueryStringW</i> (_In_ ASSOCF flags, _In_ ASSOCSTR str, _In_ LPCWSTR pszAssoc, _In_opt_ LPCWSTR pszExtra, _Out_opt_ LPWSTR pszOut, _Inout_ DWORD *pcchOut)  |
| yyi      | BOOL     | <i>ChcCmpl</i> (_In_ TCHAR w1, _In_ TCHAR w2)                                                                                                                       |
| yyi      | BOOL     | <i>ChcCmplA</i> (_In_ TCHAR w1, _In_ TCHAR w2)                                                                                                                      |
| yyi      | BOOL     | <i>ChcCmplW</i> (_In_ TCHAR w1, _In_ TCHAR w2)                                                                                                                      |
| uiiiui   | COLORREF | <i>ColorAdjustLuma</i> (COLORREF clrRGB, int n, BOOL fScale)                                                                                                        |
| uhuhuhui | COLORREF | <i>ColorHLSToRGB</i> (WORD wHue, WORD wLuminance, WORD wSaturation)                                                                                                 |
| uittti   | VOID     | <i>ColorRGBToHLS</i> (COLORREF clrRGB, _Out_ WORD *pwHue, _Out_                                                                                                     |

|         |         |                                                                                                                                                                                  |
|---------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |         | WORD *pwLuminance, _Out_ WORD *pwSaturation)                                                                                                                                     |
| ttititi | HRESULT | ConnectToConnectionPoint(_In_opt_ IUnknown *punk, _In_ REFIID riidEvent, BOOL fConnect, _In_ IUnknown *punkTarget, _Out_ DWORD *pdwCookie, _Out_opt_ IConnectionPoint **ppcpOut) |
| iwi     | HRESULT | DllInstall(BOOL bInstall, _In_opt_ PCWSTR pszCmdLine)                                                                                                                            |
| sti     | HRESULT | GetAcceptLanguages(_Out_ LPTSTR pszLanguages, _Inout_ DWORD *pcchLanguages)                                                                                                      |
| ati     | HRESULT | GetAcceptLanguagesA(_Out_ LPSTR pszLanguages, _Inout_ DWORD *pcchLanguages)                                                                                                      |
| wti     | HRESULT | GetAcceptLanguagesW(_Out_ LPWSTR pszLanguages, _Inout_ DWORD *pcchLanguages)                                                                                                     |
| tuii    | int     | GetMenuPosFromID(_In_ HMENU hmenu, UINT id)                                                                                                                                      |
| tuituii | HRESULT | HashData(_In_ BYTE *pbData, DWORD cbData, _Out_ BYTE *pbHash, DWORD cbHash)                                                                                                      |
| issii   | BOOL    | IndStrEqWorker(_In_ BOOL fCaseSens, _In_ LPCTSTR pszStr1, _In_ LPCTSTR pszStr2, _In_ int nChar)                                                                                  |
| iaaaii  | BOOL    | IndStrEqWorkerA(_In_ BOOL fCaseSens, _In_ LPCSTR pszStr1, _In_ LPCSTR pszStr2, _In_ int nChar)                                                                                   |
| iwwii   | BOOL    | IndStrEqWorkerW(_In_ BOOL fCaseSens, _In_ LPCWSTR pszStr1, _In_ LPCWSTR pszStr2, _In_ int nChar)                                                                                 |
| yi      | BOOL    | IsCharSpace(_In_ TCHAR wch)                                                                                                                                                      |
| yi      | BOOL    | IsCharSpaceA(_In_ TCHAR wch)                                                                                                                                                     |
| yi      | BOOL    | IsCharSpaceW(_In_ TCHAR wch)                                                                                                                                                     |
| i       | BOOL    | IsInternetESCEnabled(void)                                                                                                                                                       |
| uii     | BOOL    | IsOS(_In_ DWORD dwOS)                                                                                                                                                            |
| ttuii   | HRESULT | IStream_Copy(_In_ IStream *pstmFrom, _In_ IStream *pstmTo, _In_ DWORD cb)                                                                                                        |
| ttuii   | HRESULT | IStream_Read(_In_ IStream *pstm, _Out_ VOID *pv, _In_ ULONG cb)                                                                                                                  |
| tii     | HRESULT | IStream_ReadPidl(_In_ IStream* pstm, _Out_ PIDLIST_RELATIVE* ppidlOut)                                                                                                           |
| twi     | HRESULT | IStream_ReadStr(_In_ IStream *pstm, _Out_ PWSTR *ppsz)                                                                                                                           |
| ti      | HRESULT | IStream_Reset(_In_ IStream *pstm)                                                                                                                                                |
| tii     | HRESULT | IStream_Size(_In_ IStream *pstm, _Out_ ULARGE_INTEGER *pui)                                                                                                                      |
| ttuii   | HRESULT | IStream_Write(_In_ IStream *pstm, _In_ const void *pv, _In_ ULONG cb)                                                                                                            |
| tii     | HRESULT | IStream_WritePidl(_In_ IStream* pstm, _In_ PCUIDLIST_RELATIVE pidlWrite)                                                                                                         |
| twi     | HRESULT | IStream_WriteStr(_In_ IStream *pstm, _In_ PCWSTR psz)                                                                                                                            |
| ti      | VOID    | IUnknown_AtomicRelease(_Inout_opt_ void **ppunk)                                                                                                                                 |

|       |           |                                                                                                             |
|-------|-----------|-------------------------------------------------------------------------------------------------------------|
| ttti  | HRESULT   | IUnknown_GetSite(_In_ IUnknown *punk, _In_ REFIID riid, _Out_ VOID **ppvSite)                               |
| tii   | HRESULT   | IUnknown_GetWindow(_In_ IUnknown *punk, _Out_ HWND *phwnd)                                                  |
| tttti | HRESULT   | IUnknown_QueryService(_In_ IUnknown *punk, _In_ REFGUID guidService, _In_ REFIID riid, _Out_ void **ppvOut) |
| tii   | VOID      | IUnknown_Set(_Inout_ IUnknown **ppunk, _In_opt_ IUnknown *punk)                                             |
| tii   | HRESULT   | IUnknown_SetSite(_In_ IUnknown *punk, _In_ IUnknown *punkSite)                                              |
| ti    | BOOL      | MLFreeLibrary(_In_ HMODULE hModule)                                                                         |
| stuit | HINSTANCE | MLLoadLibrary(_In_ LPCTSTR lpszLibFileName, _In_ HMODULE hModule, _In_ DWORD dwCrossCodePage)               |
| atuit | HINSTANCE | MLLoadLibraryA(_In_ LPCSTR lpszLibFileName, _In_ HMODULE hModule, _In_ DWORD dwCrossCodePage)               |
| wtuit | HINSTANCE | MLLoadLibraryW(_In_ LPCWSTR lpszLibFileName, _In_ HMODULE hModule, _In_ DWORD dwCrossCodePage)              |
| sti   | HRESULT   | ParseURL(_In_ LPCTSTR pcszUrl, _Inout_ PARSEDURL *ppu)                                                      |
| ati   | HRESULT   | ParseURLA(_In_ LPCSTR pcszUrl, _Inout_ PARSEDURL *ppu)                                                      |
| wti   | HRESULT   | ParseURLW(_In_ LPCWSTR pcszUrl, _Inout_ PARSEDURL *ppu)                                                     |
| ss    | LPTSTR    | PathAddBackslash(_Inout_ LPTSTR lpszPath)                                                                   |
| aa    | LPSTR     | PathAddBackslashA(_Inout_ LPSTR lpszPath)                                                                   |
| ww    | LPWSTR    | PathAddBackslashW(_Inout_ LPWSTR lpszPath)                                                                  |
| ssi   | BOOL      | PathAddExtension(_Inout_ LPTSTR pszPath, _In_opt_ LPCTSTR pszExtension)                                     |
| aa    | BOOL      | PathAddExtensionA(_Inout_ LPSTR pszPath, _In_opt_ LPCSTR pszExtension)                                      |
| wwi   | BOOL      | PathAddExtensionW(_Inout_ LPWSTR pszPath, _In_opt_ LPCWSTR pszExtension)                                    |
| ssi   | BOOL      | PathAppend(_Inout_ LPTSTR pszPath, _In_ LPCTSTR pszMore)                                                    |
| aa    | BOOL      | PathAppendA(_Inout_ LPSTR pszPath, _In_ LPCSTR pszMore)                                                     |
| wwi   | BOOL      | PathAppendW(_Inout_ LPWSTR pszPath, _In_ LPCWSTR pszMore)                                                   |
| sis   | LPTSTR    | PathBuildRoot(_Out_ LPTSTR szRoot, _In_ int iDrive)                                                         |
| aia   | LPSTR     | PathBuildRootA(_Out_ LPSTR szRoot, _In_ int iDrive)                                                         |
| wiw   | LPWSTR    | PathBuildRootW(_Out_ LPWSTR szRoot, _In_ int iDrive)                                                        |
| ssi   | BOOL      | PathCanonicalize(_Out_ LPTSTR lpszDst, _In_ LPCTSTR lpszSrc)                                                |
| aa    | BOOL      | PathCanonicalizeA(_Out_ LPSTR lpszDst, _In_ LPCSTR lpszSrc)                                                 |
| wwi   | BOOL      | PathCanonicalizeW(_Out_ LPWSTR lpszDst, _In_ LPCWSTR lpszSrc)                                               |
| ssss  | LPTSTR    | PathCombine(_Out_ LPTSTR pszPathOut, _In_opt_ LPCTSTR pszPathIn, _In_ LPCTSTR pszMore)                      |

|         |         |                                                                                                          |
|---------|---------|----------------------------------------------------------------------------------------------------------|
| aaaa    | LPSTR   | PathCombineA(_Out_ LPSTR pszPathOut, _In_opt_ LPCSTR pszPathIn, _In_ LPCSTR pszMore)                     |
| wwwwww  | LPWSTR  | PathCombineW(_Out_ LPWSTR pszPathOut, _In_opt_ LPCWSTR pszPathIn, _In_ LPCWSTR pszMore)                  |
| sssi    | int     | PathCommonPrefix(_In_ LPCTSTR pszFile1, _In_ LPCTSTR pszFile2, _Out_opt_ LPTSTR pszPath)                 |
| aaai    | int     | PathCommonPrefixA(_In_ LPCSTR pszFile1, _In_ LPCSTR pszFile2, _Out_opt_ LPSTR pszPath)                   |
| wwwwi   | int     | PathCommonPrefixW(_In_ LPCWSTR pszFile1, _In_ LPCWSTR pszFile2, _Out_opt_ LPWSTR pszPath)                |
| tsuii   | BOOL    | PathCompactPath(_In_ HDC hDC, _Inout_ LPTSTR lpszPath, _In_ UINT dx)                                     |
| tauii   | BOOL    | PathCompactPathA(_In_ HDC hDC, _Inout_ LPSTR lpszPath, _In_ UINT dx)                                     |
| ssuiiii | BOOL    | PathCompactPathEx(_Out_ LPTSTR pszOut, _In_ LPCTSTR pszSrc, _In_ UINT cchMax, _Reserved_ DWORD dwFlags)  |
| aauiiii | BOOL    | PathCompactPathExA(_Out_ LPSTR pszOut, _In_ LPCSTR pszSrc, _In_ UINT cchMax, _Reserved_ DWORD dwFlags)   |
| wwuiiii | BOOL    | PathCompactPathExW(_Out_ LPWSTR pszOut, _In_ LPCWSTR pszSrc, _In_ UINT cchMax, _Reserved_ DWORD dwFlags) |
| twuii   | BOOL    | PathCompactPathW(_In_ HDC hDC, _Inout_ LPWSTR lpszPath, _In_ UINT dx)                                    |
| sstuii  | HRESULT | PathCreateFromUrl(_In_ PCTSTR pszUrl, _Out_ PTSTR pszPath, _Inout_ DWORD *pchPath, DWORD dwFlags)        |
| aatuii  | HRESULT | PathCreateFromUrlA(_In_ PCTSTR pszUrl, _Out_ PTSTR pszPath, _Inout_ DWORD *pchPath, DWORD dwFlags)       |
| wwuii   | HRESULT | PathCreateFromUrlAlloc(_In_ PCWSTR pszIn, _Out_ PWSTR *ppszOut, DWORD dwFlags)                           |
| wwtuii  | HRESULT | PathCreateFromUrlW(_In_ PCTSTR pszUrl, _Out_ PTSTR pszPath, _Inout_ DWORD *pchPath, DWORD dwFlags)       |
| si      | BOOL    | PathFileExists(_In_ LPCTSTR pszPath)                                                                     |
| ai      | BOOL    | PathFileExistsA(_In_ LPCSTR pszPath)                                                                     |
| wi      | BOOL    | PathFileExistsW(_In_ LPCWSTR pszPath)                                                                    |
| ss      | PTSTR   | PathFindExtension(_In_ PTSTR pszPath)                                                                    |
| aa      | PTSTR   | PathFindExtensionA(_In_ PTSTR pszPath)                                                                   |
| ww      | PTSTR   | PathFindExtensionW(_In_ PTSTR pszPath)                                                                   |
| ss      | PTSTR   | PathFindFileName(_In_ PTSTR pPath)                                                                       |
| aa      | PTSTR   | PathFindFileNameA(_In_ PTSTR pPath)                                                                      |
| ww      | PTSTR   | PathFindFileNameW(_In_ PTSTR pPath)                                                                      |
| ss      | PTSTR   | PathFindNextComponent(_In_ PTSTR pszPath)                                                                |

|      |         |                                                                                                 |
|------|---------|-------------------------------------------------------------------------------------------------|
| aa   | PTSTR   | PathFindNextComponentA(_In_ PTSTR pszPath)                                                      |
| ww   | PTSTR   | PathFindNextComponentW(_In_ PTSTR pszPath)                                                      |
| ssi  | BOOL    | PathFindOnPath(_Inout_ LPCTSTR pszFile, _In_opt_ LPCTSTR *ppszOtherDirs)                        |
| aai  | BOOL    | PathFindOnPathA(_Inout_ LPSTR pszFile, _In_opt_ LPCSTR *ppszOtherDirs)                          |
| wwi  | BOOL    | PathFindOnPathW(_Inout_ LPWSTR pszFile, _In_opt_ LPCWSTR *ppszOtherDirs)                        |
| ssis | LPCTSTR | PathFindSuffixArray(_In_ LPCTSTR pszPath, _In_ const LPCTSTR *apszSuffix, _In_ int iArraySize)  |
| aaia | LPCSTR  | PathFindSuffixArrayA(_In_ LPCSTR pszPath, _In_ const LPCSTR *apszSuffix, _In_ int iArraySize)   |
| wwiw | LPCWSTR | PathFindSuffixArrayW(_In_ LPCWSTR pszPath, _In_ const LPCWSTR *apszSuffix, _In_ int iArraySize) |
| ss   | PTSTR   | PathGetArgs(_In_ PTSTR pszPath)                                                                 |
| aa   | PTSTR   | PathGetArgsA(_In_ PTSTR pszPath)                                                                |
| ww   | PTSTR   | PathGetArgsW(_In_ PTSTR pszPath)                                                                |
| zui  | UINT    | PathGetCharType(_In_ TCHAR ch)                                                                  |
| zui  | UINT    | PathGetCharTypeA(_In_ TCHAR ch)                                                                 |
| zui  | UINT    | PathGetCharTypeW(_In_ TCHAR ch)                                                                 |
| si   | int     | PathGetDriveNumber(_In_ LPCTSTR lpsz)                                                           |
| ai   | int     | PathGetDriveNumberA(_In_ LPCSTR lpsz)                                                           |
| wi   | int     | PathGetDriveNumberW(_In_ LPCWSTR lpsz)                                                          |
| ssi  | BOOL    | PathIsContentType(_In_ LPCTSTR pszPath, _In_ LPCTSTR pszContentType)                            |
| aai  | BOOL    | PathIsContentTypeA(_In_ LPCSTR pszPath, _In_ LPCSTR pszContentType)                             |
| wwi  | BOOL    | PathIsContentTypeW(_In_ LPCWSTR pszPath, _In_ LPCWSTR pszContentType)                           |
| si   | BOOL    | PathIsDirectory(_In_ LPCTSTR pszPath)                                                           |
| ai   | BOOL    | PathIsDirectoryA(_In_ LPCSTR pszPath)                                                           |
| si   | BOOL    | PathIsDirectoryEmpty(_In_ LPCTSTR pszPath)                                                      |
| ai   | BOOL    | PathIsDirectoryEmptyA(_In_ LPCSTR pszPath)                                                      |
| wi   | BOOL    | PathIsDirectoryEmptyW(_In_ LPCWSTR pszPath)                                                     |
| wi   | BOOL    | PathIsDirectoryW(_In_ LPCWSTR pszPath)                                                          |
| si   | BOOL    | PathIsFileSpec(_In_ LPCTSTR lpszPath)                                                           |
| ai   | BOOL    | PathIsFileSpecA(_In_ LPCSTR lpszPath)                                                           |

|      |      |                                                                   |
|------|------|-------------------------------------------------------------------|
| wi   | BOOL | PathIsFileSpecW(_In_ LPCWSTR lpszPath)                            |
| si   | BOOL | PathIsLPNFileSpec(_In_ LPCTSTR pszName)                           |
| ai   | BOOL | PathIsLPNFileSpecA(_In_ LPCSTR pszName)                           |
| wi   | BOOL | PathIsLPNFileSpecW(_In_ LPCWSTR pszName)                          |
| si   | BOOL | PathIsNetworkPath(_In_ LPCTSTR pszPath)                           |
| ai   | BOOL | PathIsNetworkPathA(_In_ LPCSTR pszPath)                           |
| wi   | BOOL | PathIsNetworkPathW(_In_ LPCWSTR pszPath)                          |
| ssi  | BOOL | PathIsPrefix(_In_ IN LPCTSTR pszPrefix, _In_ IN LPCTSTR pszPath)  |
| aai  | BOOL | PathIsPrefixA(_In_ IN LPCSTR pszPrefix, _In_ IN LPCSTR pszPath)   |
| wwi  | BOOL | PathIsPrefixW(_In_ IN LPCWSTR pszPrefix, _In_ IN LPCWSTR pszPath) |
| si   | BOOL | PathIsRelative(_In_ LPCTSTR lpszPath)                             |
| ai   | BOOL | PathIsRelativeA(_In_ LPCSTR lpszPath)                             |
| wi   | BOOL | PathIsRelativeW(_In_ LPCWSTR lpszPath)                            |
| si   | BOOL | PathIsRoot(_In_ LPCTSTR pPath)                                    |
| ai   | BOOL | PathIsRootA(_In_ LPCSTR pPath)                                    |
| wi   | BOOL | PathIsRootW(_In_ LPCWSTR pPath)                                   |
| ssi  | BOOL | PathIsSameRoot(_In_ LPCTSTR pszPath1, _In_ LPCTSTR pszPath2)      |
| aai  | BOOL | PathIsSameRootA(_In_ LPCSTR pszPath1, _In_ LPCSTR pszPath2)       |
| wwi  | BOOL | PathIsSameRootW(_In_ LPCWSTR pszPath1, _In_ LPCWSTR pszPath2)     |
| suii | BOOL | PathIsSystemFolder(_In_opt_ LPCTSTR pszPath, _In_ DWORD dwAttrb)  |
| auii | BOOL | PathIsSystemFolderA(_In_opt_ LPCSTR pszPath, _In_ DWORD dwAttrb)  |
| wuii | BOOL | PathIsSystemFolderW(_In_opt_ LPCWSTR pszPath, _In_ DWORD dwAttrb) |
| si   | BOOL | PathIsUNC(_In_ LPCTSTR pszPath)                                   |
| ai   | BOOL | PathIsUNCA(_In_ LPCSTR pszPath)                                   |
| si   | BOOL | PathIsUNCServer(_In_ LPCTSTR pszPath)                             |
| ai   | BOOL | PathIsUNCServerA(_In_ LPCSTR pszPath)                             |
| si   | BOOL | PathIsUNCServerShare(_In_ LPCTSTR pszPath)                        |
| ai   | BOOL | PathIsUNCServerShareA(_In_ LPCSTR pszPath)                        |
| wi   | BOOL | PathIsUNCServerShareW(_In_ LPCWSTR pszPath)                       |
| wi   | BOOL | PathIsUNCServerW(_In_ LPCWSTR pszPath)                            |
| wi   | BOOL | PathIsUNCW(_In_ LPCWSTR pszPath)                                  |

|          |         |                                                                                                                                 |
|----------|---------|---------------------------------------------------------------------------------------------------------------------------------|
| si       | BOOL    | PathIsURL(_In_ LPCTSTR pszPath)                                                                                                 |
| ai       | BOOL    | PathIsURLA(_In_ LPCSTR pszPath)                                                                                                 |
| wi       | BOOL    | PathIsURLW(_In_ LPCWSTR pszPath)                                                                                                |
| si       | BOOL    | PathMakePretty(_Inout_ LPTSTR lpPath)                                                                                           |
| ai       | BOOL    | PathMakePrettyA(_Inout_ LPSTR lpPath)                                                                                           |
| wi       | BOOL    | PathMakePrettyW(_Inout_ LPWSTR lpPath)                                                                                          |
| si       | BOOL    | PathMakeSystemFolder(_In_ LPTSTR pszPath)                                                                                       |
| ai       | BOOL    | PathMakeSystemFolderA(_In_ LPSTR pszPath)                                                                                       |
| wi       | BOOL    | PathMakeSystemFolderW(_In_ LPWSTR pszPath)                                                                                      |
| aa       | BOOL    | PathMatchSpec(_In_ LPCSTR pszFile, _In_ LPCSTR pszSpec)                                                                         |
| aa       | BOOL    | PathMatchSpecA(_In_ LPCSTR pszFile, _In_ LPCSTR pszSpec)                                                                        |
| ssuii    | HRESULT | PathMatchSpecEx(_In_ LPCTSTR pszFile, _In_ LPCTSTR pszSpec, _In_ DWORD dwFlags)                                                 |
| aauii    | HRESULT | PathMatchSpecExA(_In_ LPCSTR pszFile, _In_ LPCSTR pszSpec, _In_ DWORD dwFlags)                                                  |
| wwuii    | HRESULT | PathMatchSpecExW(_In_ LPCWSTR pszFile, _In_ LPCWSTR pszSpec, _In_ DWORD dwFlags)                                                |
| aa       | BOOL    | PathMatchSpecW(_In_ LPCSTR pszFile, _In_ LPCSTR pszSpec)                                                                        |
| si       | int     | PathParseIconLocation(_Inout_ LPTSTR pszIconFile)                                                                               |
| ai       | int     | PathParseIconLocationA(_Inout_ LPSTR pszIconFile)                                                                               |
| wi       | int     | PathParseIconLocationW(_Inout_ LPWSTR pszIconFile)                                                                              |
| si       | BOOL    | PathQuoteSpaces(_Inout_ LPTSTR lpsz)                                                                                            |
| ai       | BOOL    | PathQuoteSpacesA(_Inout_ LPSTR lpsz)                                                                                            |
| wi       | BOOL    | PathQuoteSpacesW(_Inout_ LPWSTR lpsz)                                                                                           |
| ssuisuii | BOOL    | PathRelativePathTo(_Out_ LPTSTR pszPath, _In_ LPCTSTR pszFrom, _In_ DWORD dwAttrFrom, _In_ LPCTSTR pszTo, _In_ DWORD dwAttrTo)  |
| aauii    | BOOL    | PathRelativePathToA(_Out_ LPSTR pszPath, _In_ LPCSTR pszFrom, _In_ DWORD dwAttrFrom, _In_ LPCSTR pszTo, _In_ DWORD dwAttrTo)    |
| wwuii    | BOOL    | PathRelativePathToW(_Out_ LPWSTR pszPath, _In_ LPCWSTR pszFrom, _In_ DWORD dwAttrFrom, _In_ LPCWSTR pszTo, _In_ DWORD dwAttrTo) |
| si       | VOID    | PathRemoveArgs(_Inout_ LPTSTR pszPath)                                                                                          |
| ai       | VOID    | PathRemoveArgsA(_Inout_ LPSTR pszPath)                                                                                          |
| wi       | VOID    | PathRemoveArgsW(_Inout_ LPWSTR pszPath)                                                                                         |
| ss       | LPTSTR  | PathRemoveBackslash(_Inout_ LPTSTR lpszPath)                                                                                    |
|          |         |                                                                                                                                 |

|       |        |                                                                                                                   |
|-------|--------|-------------------------------------------------------------------------------------------------------------------|
| aa    | LPSTR  | PathRemoveBackslashA(_Inout_ LPSTR lpszPath)                                                                      |
| ww    | LPWSTR | PathRemoveBackslashW(_Inout_ LPWSTR lpszPath)                                                                     |
| si    | VOID   | PathRemoveBlanks(_Inout_ LPTSTR lpszString)                                                                       |
| ai    | VOID   | PathRemoveBlanksA(_Inout_ LPSTR lpszString)                                                                       |
| wi    | VOID   | PathRemoveBlanksW(_Inout_ LPWSTR lpszString)                                                                      |
| si    | VOID   | PathRemoveExtension(_Inout_ LPTSTR pszPath)                                                                       |
| ai    | VOID   | PathRemoveExtensionA(_Inout_ LPSTR pszPath)                                                                       |
| wi    | VOID   | PathRemoveExtensionW(_Inout_ LPWSTR pszPath)                                                                      |
| si    | BOOL   | PathRemoveFileSpec(_Inout_ LPTSTR pszPath)                                                                        |
| ai    | BOOL   | PathRemoveFileSpecA(_Inout_ LPSTR pszPath)                                                                        |
| wi    | BOOL   | PathRemoveFileSpecW(_Inout_ LPWSTR pszPath)                                                                       |
| ssi   | BOOL   | PathRenameExtension(_Inout_ LPTSTR pszPath, _In_ LPCTSTR pszExt)                                                  |
| aa    | BOOL   | PathRenameExtensionA(_Inout_ LPSTR pszPath, _In_ LPCSTR pszExt)                                                   |
| ww    | BOOL   | PathRenameExtensionW(_Inout_ LPWSTR pszPath, _In_ LPCWSTR pszExt)                                                 |
| ssuii | BOOL   | PathSearchAndQualify(_In_ LPCTSTR pcszPath, _Out_ LPTSTR pszFullyQualifiedPath, _In_ UINT cchFullyQualifiedPath)  |
| aauii | BOOL   | PathSearchAndQualifyA(_In_ LPCSTR pcszPath, _Out_ LPSTR pszFullyQualifiedPath, _In_ UINT cchFullyQualifiedPath)   |
| wwuii | BOOL   | PathSearchAndQualifyW(_In_ LPCWSTR pcszPath, _Out_ LPWSTR pszFullyQualifiedPath, _In_ UINT cchFullyQualifiedPath) |
| tiai  | VOID   | PathSetDlgItemPath(_In_ HWND hDlg, _In_ int id, _In_ LPCSTR pszPath)                                              |
| tiai  | VOID   | PathSetDlgItemPathA(_In_ HWND hDlg, _In_ int id, _In_ LPCSTR pszPath)                                             |
| tiai  | VOID   | PathSetDlgItemPathW(_In_ HWND hDlg, _In_ int id, _In_ LPCWSTR pszPath)                                            |
| ss    | PTSTR  | PathSkipRoot(_In_ PTSTR pszPath)                                                                                  |
| aa    | PTSTR  | PathSkipRootA(_In_ PTSTR pszPath)                                                                                 |
| ww    | PTSTR  | PathSkipRootW(_In_ PTSTR pszPath)                                                                                 |
| si    | VOID   | PathStripPath(_Inout_ LPTSTR pszPath)                                                                             |
| ai    | VOID   | PathStripPathA(_Inout_ LPSTR pszPath)                                                                             |
| wi    | VOID   | PathStripPathW(_Inout_ LPWSTR pszPath)                                                                            |
| si    | BOOL   | PathStripToRoot(_Inout_ LPTSTR szRoot)                                                                            |
| ai    | BOOL   | PathStripToRootA(_Inout_ LPSTR szRoot)                                                                            |
| wi    | BOOL   | PathStripToRootW(_Inout_ LPWSTR szRoot)                                                                           |

|           |          |                                                                                                                                                                       |
|-----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si        | VOID     | PathUndecorate(_Inout_ LPTSTR pszPath)                                                                                                                                |
| ai        | VOID     | PathUndecorateA(_Inout_ LPSTR pszPath)                                                                                                                                |
| wi        | VOID     | PathUndecorateW(_Inout_ LPWSTR pszPath)                                                                                                                               |
| ssuii     | BOOL     | PathUnExpandEnvStrings(_In_ LPCTSTR pszPath, _Out_ LPTSTR pszBuf, _In_ UINT cchBuf)                                                                                   |
| aauii     | BOOL     | PathUnExpandEnvStringsA(_In_ LPCSTR pszPath, _Out_ LPSTR pszBuf, _In_ UINT cchBuf)                                                                                    |
| wwuii     | BOOL     | PathUnExpandEnvStringsW(_In_ LPCWSTR pszPath, _Out_ LPWSTR pszBuf, _In_ UINT cchBuf)                                                                                  |
| si        | BOOL     | PathUnmakeSystemFolder(_In_ LPTSTR pszPath)                                                                                                                           |
| ai        | BOOL     | PathUnmakeSystemFolderA(_In_ LPSTR pszPath)                                                                                                                           |
| wi        | BOOL     | PathUnmakeSystemFolderW(_In_ LPWSTR pszPath)                                                                                                                          |
| si        | VOID     | PathUnquoteSpaces(_Inout_ LPTSTR lpsz)                                                                                                                                |
| ai        | VOID     | PathUnquoteSpacesA(_Inout_ LPSTR lpsz)                                                                                                                                |
| wi        | VOID     | PathUnquoteSpacesW(_Inout_ LPWSTR lpsz)                                                                                                                               |
| ttti      | HRESULT  | QISearch(_In_ void *that, _In_ LPCQITAB pqit, _In_ REFIID riid, _Out_ void **ppv)                                                                                     |
| tuiuit    | HANDLE   | SHAllocShared(_In_opt_ const void *pvData, _In_ DWORD dwSize, _In_ DWORD dwDestinationProcessId)                                                                      |
| awii      | int      | SHAnsiToAnsi(_In_ LPCSTR pszSrc, _Out_ LPWSTR pszDst, int cchBuf)                                                                                                     |
| atii      | int      | SHAnsiToUnicode(_In_ PCSTR pszSrc, _Out_ PWSTR pwszDst, int cwchBuf)                                                                                                  |
| tuii      | HRESULT  | SHAutoComplete(_In_ HWND hwndEdit, DWORD dwFlags)                                                                                                                     |
| tstuiui   | LSTATUS  | SHCopyKey(_In_ HKEY hkeySrc, _In_opt_ LPCTSTR pszSrcSubKey, _In_ HKEY hkeyDest, _Reserved_ DWORD fReserved)                                                           |
| tatuiui   | LSTATUS  | SHCopyKeyA(_In_ HKEY hkeySrc, _In_opt_ LPCSTR pszSrcSubKey, _In_ HKEY hkeyDest, _Reserved_ DWORD fReserved)                                                           |
| twtuiui   | LSTATUS  | SHCopyKeyW(_In_ HKEY hkeySrc, _In_opt_ LPCWSTR pszSrcSubKey, _In_ HKEY hkeyDest, _Reserved_ DWORD fReserved)                                                          |
| tuit      | IStream* | SHCreateMemStream(_In_opt_ const BYTE *pInit, _In_ UINT cbInit)                                                                                                       |
| tt        | HPALETTE | SHCreateShellPalette(_In_opt_ HDC hdc)                                                                                                                                |
| suiti     | HRESULT  | SHCreateStreamOnFile(_In_ LPCTSTR pszFile, _In_ DWORD grfMode, _Out_ IStream **ppstm)                                                                                 |
| auiti     | HRESULT  | SHCreateStreamOnFileA(_In_ LPCSTR pszFile, _In_ DWORD grfMode, _Out_ IStream **ppstm)                                                                                 |
| wuiuiitti | HRESULT  | SHCreateStreamOnFileEx(_In_ LPCWSTR pszFile, _In_ DWORD grfMode, _In_ DWORD dwAttributes, _In_ BOOL fCreate, _Reserved_ IStream *pstmTemplate, _Out_ IStream **ppstm) |

|           |         |                                                                                                                                                                                          |
|-----------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wuiti     | HRESULT | SHCreateStreamOnFileW(_In_ LPCWSTR pszFile, _In_ DWORD grfMode, _Out_ IStream **ppstm)                                                                                                   |
| ttuiti    | BOOL    | SHCreateThread(_In_ LPTHREAD_START_ROUTINE pfnThreadProc, _In_opt_ void *pData, _In_ SHCT_FLAGS dwFlags, _In_opt_ LPTHREAD_START_ROUTINE pfnCallback)                                    |
| tti       | HRESULT | SHCreateThreadRef(_In_ LONG *pcRef, _Out_ IUnknown **ppunk)                                                                                                                              |
| ttuitti   | BOOL    | SHCreateThreadWithHandle(_In_ LPTHREAD_START_ROUTINE pfnThreadProc, _In_opt_ void *pData, _In_ SHCT_FLAGS flags, _In_opt_ LPTHREAD_START_ROUTINE pfnCallback, _Out_opt_ HANDLE *pHandle) |
| tsui      | LSTATUS | SHDeleteEmptyKey(_In_ HKEY hkey, _In_opt_ LPCTSTR pszSubKey)                                                                                                                             |
| taui      | LSTATUS | SHDeleteEmptyKeyA(_In_ HKEY hkey, _In_opt_ LPCSTR pszSubKey)                                                                                                                             |
| twui      | LSTATUS | SHDeleteEmptyKeyW(_In_ HKEY hkey, _In_opt_ LPCWSTR pszSubKey)                                                                                                                            |
| tsui      | LSTATUS | SHDeleteKey(_In_ HKEY hkey, _In_opt_ LPCTSTR pszSubKey)                                                                                                                                  |
| taui      | LSTATUS | SHDeleteKeyA(_In_ HKEY hkey, _In_opt_ LPCSTR pszSubKey)                                                                                                                                  |
| twui      | LSTATUS | SHDeleteKeyW(_In_ HKEY hkey, _In_opt_ LPCWSTR pszSubKey)                                                                                                                                 |
| tssui     | LSTATUS | SHDeleteValue(HKEY hkey, LPCTSTR pszSubKey, LPCTSTR pszValue)                                                                                                                            |
| taai      | LSTATUS | SHDeleteValueA(HKEY hkey, LPCSTR pszSubKey, LPCSTR pszValue)                                                                                                                             |
| twwai     | LSTATUS | SHDeleteValueW(HKEY hkey, LPCWSTR pszSubKey, LPCWSTR pszValue)                                                                                                                           |
| ttssuitti | int     | ShellMessageBox(_In_ HINSTANCE hInst, _In_ HWND hWnd, _In_ LPCTSTR pszMsg, _In_ LPCTSTR pszTitle, _In_ UINT fuStyle, _In_ ...)                                                           |
| ttaauitti | int     | ShellMessageBoxA(_In_ HINSTANCE hInst, _In_ HWND hWnd, _In_ LPCSTR pszMsg, _In_ LPCSTR pszTitle, _In_ UINT fuStyle, _In_ ...)                                                            |
| ttwwuitti | int     | ShellMessageBoxW(_In_ HINSTANCE hInst, _In_ HWND hWnd, _In_ LPCWSTR pszMsg, _In_ LPCWSTR pszTitle, _In_ UINT fuStyle, _In_ ...)                                                          |
| tui       | LSTATUS | SHEnumKeyEx(_In_ HKEY hkey, DWORD dwIndex, LPCTSTR pszName, _In_ LPDWORD pcchName)                                                                                                       |
| tuiatui   | LSTATUS | SHEnumKeyExA(_In_ HKEY hkey, DWORD dwIndex, LPCTSTR pszName, _In_ LPDWORD pcchName)                                                                                                      |
| tuiwtui   | LSTATUS | SHEnumKeyExW(_In_ HKEY hkey, DWORD dwIndex, LPWSTR pszName, _In_ LPDWORD pcchName)                                                                                                       |
| tui       | LSTATUS | SHEnumValue(_In_ HKEY hkey, _In_ DWORD dwIndex, LPCTSTR pszValueName, _Inout_opt_ LPDWORD pcchValueName, _Out_opt_ LPDWORD pdwType, LPVOID pvData, _Inout_opt_ LPDWORD pcbData)          |
|           |         | SHEnumValueA(_In_ HKEY hkey, _In_ DWORD dwIndex, LPCTSTR                                                                                                                                 |

|            |         |                                                                                                                                                                                 |
|------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiattttui | LSTATUS | pszValueName, _Inout_opt_ LPDWORD pcchValueName, _Out_opt_ LPDWORD pdwType, LPVOID pvData, _Inout_opt_ LPDWORD pcbData)                                                         |
| tuiwttttui | LSTATUS | SHEnumValueW(_In_ HKEY hkey, _In_ DWORD dwIndex, LPWSTR pszValueName, _Inout_opt_ LPDWORD pcchValueName, _Out_opt_ LPDWORD pdwType, LPVOID pvData, _Inout_opt_ LPDWORD pcbData) |
| ttsuii     | int     | SHFormatDateTime(_In_ const FILETIME UNALIGNED *pft, _Inout_opt_ DWORD *pdwFlags, _Out_ LPTSTR pszBuf, UINT cchBuf)                                                             |
| ttauui     | int     | SHFormatDateTimeA(_In_ const FILETIME UNALIGNED *pft, _Inout_opt_ DWORD *pdwFlags, _Out_ LPSTR pszBuf, UINT cchBuf)                                                             |
| ttwuii     | int     | SHFormatDateTimeW(_In_ const FILETIME UNALIGNED *pft, _Inout_opt_ DWORD *pdwFlags, _Out_ LPWSTR pszBuf, UINT cchBuf)                                                            |
| tuii       | BOOL    | SHFreeShared(_In_ HANDLE hData, _In_ DWORD dwProcessId)                                                                                                                         |
| tuii       | HRESULT | SHGetInverseCMAP(_Out_ BYTE *pbMap, _In_ ULONG cbMap)                                                                                                                           |
| ti         | HRESULT | SHGetThreadRef(_In_ IUnknown **ppunk)                                                                                                                                           |
| tsstttui   | LSTATUS | SHGetValue(_In_ HKEY hkey, _In_opt_ LPCTSTR pszSubKey, _In_opt_ LPCTSTR pszValue, _Out_opt_ LPDWORD pdwType, _Out_opt_ LPVOID pvData, _Inout_opt_ LPDWORD pcbData)              |
| taatttui   | LSTATUS | SHGetValueA(_In_ HKEY hkey, _In_opt_ LPCSTR pszSubKey, _In_opt_ LPCSTR pszValue, _Out_opt_ LPDWORD pdwType, _Out_opt_ LPVOID pvData, _Inout_opt_ LPDWORD pcbData)               |
| twwtttui   | LSTATUS | SHGetValueW(_In_ HKEY hkey, _In_opt_ LPCWSTR pszSubKey, _In_opt_ LPCWSTR pszValue, _Out_opt_ LPDWORD pdwType, _Out_opt_ LPVOID pvData, _Inout_opt_ LPDWORD pcbData)             |
| twuitti    | HRESULT | SHGetViewStatePropertyBag(_In_opt_ PCIDLIST_ABSOLUTE pidl, _In_opt_ PCWSTR pszBagName, DWORD dwFlags, _In_ REFIID riid, _Out_ void **ppv)                                       |
| tti        | HRESULT | SHIsChildOrSelf(_In_ HWND hWndParent, _In_ HWND hWnd)                                                                                                                           |
| wtuiti     | HRESULT | SHLoadIndirectString(_In_ PCWSTR pszSource, _Out_ PWSTR pszOutBuf, _In_ UINT cchOutBuf, _Reserved_ void **ppvReserved)                                                          |
| tuit       | void*   | SHLockShared(_In_ HANDLE hData, _In_ DWORD dwProcessId)                                                                                                                         |
| tssuiisi   | int     | SHMessageBoxCheck(_In_opt_ HWND hwnd, _In_ LPCTSTR pszText, _In_ LPCTSTR pszCaption, UINT uType, int iDefault, _In_ LPCTSTR pszRegVal)                                          |
| taauiaai   | int     | SHMessageBoxCheckA(_In_opt_ HWND hwnd, _In_ LPCSTR pszText, _In_ LPCSTR pszCaption, UINT uType, int iDefault, _In_ LPCSTR pszRegVal)                                            |
| twwuiiwi   | int     | SHMessageBoxCheckW(_In_opt_ HWND hwnd, _In_ LPCWSTR pszText, _In_ LPCWSTR pszCaption, UINT uType, int iDefault, _In_ LPCWSTR pszRegVal)                                         |

|           |          |                                                                                                                                                                     |
|-----------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tssuit    | IStream* | SHOpenRegStream(_In_ HKEY hkey, _In_opt_ LPCTSTR pszSubkey, _In_opt_ LPCTSTR pszValue, _In_ DWORD grfMode)                                                          |
| tssuit    | IStream* | SHOpenRegStream2(_In_ HKEY hkey, _In_opt_ LPCTSTR pszSubkey, _In_opt_ LPCTSTR pszValue, _In_ DWORD grfMode)                                                         |
| taauit    | IStream* | SHOpenRegStream2A(_In_ HKEY hkey, _In_opt_ LPCSTR pszSubkey, _In_opt_ LPCSTR pszValue, _In_ DWORD grfMode)                                                          |
| twwwuit   | IStream* | SHOpenRegStream2W(_In_ HKEY hkey, _In_opt_ LPCWSTR pszSubkey, _In_opt_ LPCWSTR pszValue, _In_ DWORD grfMode)                                                        |
| taauit    | IStream* | SHOpenRegStreamA(_In_ HKEY hkey, _In_opt_ LPCSTR pszSubkey, _In_opt_ LPCSTR pszValue, _In_ DWORD grfMode)                                                           |
| twwwuit   | IStream* | SHOpenRegStreamW(_In_ HKEY hkey, _In_opt_ LPCWSTR pszSubkey, _In_opt_ LPCWSTR pszValue, _In_ DWORD grfMode)                                                         |
| ttttui    | LSTATUS  | SHQueryInfoKey(_In_ HKEY hkey, _Out_opt_ LPDWORD pcSubKeys, _Out_opt_ LPDWORD pcchMaxSubKeyLen, _Out_opt_ LPDWORD pcValues, _Out_opt_ LPDWORD pcchMaxValueNameLen)  |
| ttttui    | LSTATUS  | SHQueryInfoKeyA(_In_ HKEY hkey, _Out_opt_ LPDWORD pcSubKeys, _Out_opt_ LPDWORD pcchMaxSubKeyLen, _Out_opt_ LPDWORD pcValues, _Out_opt_ LPDWORD pcchMaxValueNameLen) |
| ttttui    | LSTATUS  | SHQueryInfoKeyW(_In_ HKEY hkey, _Out_opt_ LPDWORD pcSubKeys, _Out_opt_ LPDWORD pcchMaxSubKeyLen, _Out_opt_ LPDWORD pcValues, _Out_opt_ LPDWORD pcchMaxValueNameLen) |
| tsTTTTui  | DWORD    | SHQueryValueEx(_In_ HKEY hkey, _In_opt_ LPCTSTR pszValue, LPDWORD pdwReserved, _Out_opt_ LPDWORD pdwType, _Out_opt_ LPVOID pvData, _Inout_opt_ LPDWORD pcbData)     |
| tattttui  | DWORD    | SHQueryValueExA(_In_ HKEY hkey, _In_opt_ LPCSTR pszValue, LPDWORD pdwReserved, _Out_opt_ LPDWORD pdwType, _Out_opt_ LPVOID pvData, _Inout_opt_ LPDWORD pcbData)     |
| twTTTTui  | DWORD    | SHQueryValueExW(_In_ HKEY hkey, _In_opt_ LPCWSTR pszValue, LPDWORD pdwReserved, _Out_opt_ LPDWORD pdwType, _Out_opt_ LPVOID pvData, _Inout_opt_ LPDWORD pcbData)    |
| tui       | LSTATUS  | SHRegCloseUSKey(_In_ HUSKEY hUSKey)                                                                                                                                 |
| suittuiui | LSTATUS  | SHRegCreateUSKey(_In_ LPCTSTR pszPath, _In_ REGSAM samDesired, _In_opt_ HUSKEY hRelativeUSKey, _Out_ PHUSKEY phNewUSKey, _In_ DWORD dwFlags)                        |
| auittuiui | LSTATUS  | SHRegCreateUSKeyA(_In_ LPCSTR pszPath, _In_ REGSAM samDesired, _In_opt_ HUSKEY hRelativeUSKey, _Out_ PHUSKEY phNewUSKey, _In_ DWORD dwFlags)                        |
| wuittuiui | LSTATUS  | SHRegCreateUSKeyW(_In_ LPCWSTR pszPath, _In_ REGSAM samDesired, _In_opt_ HUSKEY hRelativeUSKey, _Out_ PHUSKEY phNewUSKey, _In_ DWORD dwFlags)                       |
| tauiui    | LSTATUS  | SHRegDeleteEmptyUSKey(_In_ HUSKEY hUSKey, _In_ LPCSTR pszValue, _In_ SHREGDEL_FLAGS delRegFlags)                                                                    |
| tauiui    | LSTATUS  | SHRegDeleteEmptyUSKeyA(_In_ HUSKEY hUSKey, _In_ LPCSTR                                                                                                              |



|           |         |                                                                                                                                                                                                                                      |
|-----------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |         | LPCTSTR pszValue, _Out_ LPTSTR pszPath, DWORD dwFlags)                                                                                                                                                                               |
| taauiui   | LSTATUS | SHRegGetPathA(_In_ HKEY hkey, _In_ LPCSTR pszSubkey, _In_ LPCSTR pszValue, _Out_ LPSTR pszPath, DWORD dwFlags)                                                                                                                       |
| twwuiui   | LSTATUS | SHRegGetPathW(_In_ HKEY hkey, _In_ LPCWSTR pszSubkey, _In_ LPCWSTR pszValue, _Out_ LPWSTR pszPath, DWORD dwFlags)                                                                                                                    |
| ssttitiui | LSTATUS | SHRegGetValue(_In_ LPCTSTR pszSubKey, _In_opt_ LPCTSTR pszValue, _Inout_opt_ DWORD *pdwType, _Out_opt_ void *pvData, _Inout_opt_ DWORD *pcbData, _In_ BOOL fIgnoreHKCU, _In_opt_ void *pvDefaultData, _In_ DWORD dwDefaultDataSize)  |
| aattitiui | LSTATUS | SHRegGetValueA(_In_ LPCSTR pszSubKey, _In_opt_ LPCSTR pszValue, _Inout_opt_ DWORD *pdwType, _Out_opt_ void *pvData, _Inout_opt_ DWORD *pcbData, _In_ BOOL fIgnoreHKCU, _In_opt_ void *pvDefaultData, _In_ DWORD dwDefaultDataSize)   |
| wwtitiui  | LSTATUS | SHRegGetValueW(_In_ LPCWSTR pszSubKey, _In_opt_ LPCWSTR pszValue, _Inout_opt_ DWORD *pdwType, _Out_opt_ void *pvData, _Inout_opt_ DWORD *pcbData, _In_ BOOL fIgnoreHKCU, _In_opt_ void *pvDefaultData, _In_ DWORD dwDefaultDataSize) |
| tssittui  | LSTATUS | SHRegGetValue(_In_ HKEY hkey, _In_ LPCTSTR pszSubKey, _In_ LPCTSTR pszValue, _In_ SRRF srrfFlags, _Inout_ LPDWORD pdwType, _Out_ LPVOID pvData, _Inout_ LPDWORD pcbData)                                                             |
| taaittui  | LSTATUS | SHRegGetValueA(_In_ HKEY hkey, _In_ LPCSTR pszSubKey, _In_ LPCSTR pszValue, _In_ SRRF srrfFlags, _Inout_ LPDWORD pdwType, _Out_ LPVOID pvData, _Inout_ LPDWORD pcbData)                                                              |
| twwittui  | LSTATUS | SHRegGetValueW(_In_ HKEY hkey, _In_ LPCWSTR pszSubKey, _In_ LPCWSTR pszValue, _In_ SRRF srrfFlags, _Inout_ LPDWORD pdwType, _Out_ LPVOID pvData, _Inout_ LPDWORD pcbData)                                                            |
| suittui   | LSTATUS | SHRegOpenUSKey(_In_ LPCTSTR pszPath, _In_ REGSAM samDesired, _In_opt_ HUSKEY hRelativeUSKey, _Out_ PHUSKEY phNewUSKey, _In_ BOOL fIgnoreHKCU)                                                                                        |
| auittui   | LSTATUS | SHRegOpenUSKeyA(_In_ LPCSTR pszPath, _In_ REGSAM samDesired, _In_opt_ HUSKEY hRelativeUSKey, _Out_ PHUSKEY phNewUSKey, _In_ BOOL fIgnoreHKCU)                                                                                        |
| wuittui   | LSTATUS | SHRegOpenUSKeyW(_In_ LPCWSTR pszPath, _In_ REGSAM samDesired, _In_opt_ HUSKEY hRelativeUSKey, _Out_ PHUSKEY phNewUSKey, _In_ BOOL fIgnoreHKCU)                                                                                       |
| ttttuiui  | LSTATUS | SHRegQueryInfoUSKey(_In_ HUSKEY hUSKey, _Out_opt_ LPDWORD pcSubKeys, _Out_opt_ LPDWORD pcchMaxSubKeyLen, _Out_opt_ LPDWORD pcValues, _Out_opt_ LPDWORD pcchMaxValueNameLen, _In_ SHREGENUM_FLAGS enumRegFlags)                       |
| ttttuiui  | LSTATUS | SHRegQueryInfoUSKeyA(_In_ HUSKEY hUSKey, _Out_opt_ LPDWORD pcSubKeys, _Out_opt_ LPDWORD pcchMaxSubKeyLen, _Out_opt_ LPDWORD pcValues, _Out_opt_ LPDWORD pcchMaxValueNameLen, _In_ SHREGENUM_FLAGS enumRegFlags)                      |
|           |         | SHRegQueryInfoUSKeyW(_In_ HUSKEY hUSKey, _Out_opt_                                                                                                                                                                                   |

|            |         |                                                                                                                                                                                                                                              |
|------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tttttiii   | LSTATUS | LPDWORD pcSubKeys, _Out_opt_ LPDWORD pcchMaxSubKeyLen, _Out_opt_ LPDWORD pcValues, _Out_opt_ LPDWORD pcchMaxValueNameLen, _In_ SHREGENUM_FLAGS enumRegFlags)                                                                                 |
| tsittitiii | LSTATUS | SHRegQueryUSValue(_In_ HUSKEY hUSKey, _In_opt_ LPCTSTR pszValue, _Inout_opt_ LPDWORD *pdwType, _Out_opt_ LPVOID *pvData, _Inout_ LPDWORD *pcbData, _In_ BOOL fIgnoreHKCU, _In_opt_ LPVOID *pvDefaultData, _In_opt_ DWORD dwDefaultDataSize)  |
| tatttitiii | LSTATUS | SHRegQueryUSValueA(_In_ HUSKEY hUSKey, _In_opt_ LPCSTR pszValue, _Inout_opt_ LPDWORD *pdwType, _Out_opt_ LPVOID *pvData, _Inout_ LPDWORD *pcbData, _In_ BOOL fIgnoreHKCU, _In_opt_ LPVOID *pvDefaultData, _In_opt_ DWORD dwDefaultDataSize)  |
| twtttitiii | LSTATUS | SHRegQueryUSValueW(_In_ HUSKEY hUSKey, _In_opt_ LPCWSTR pszValue, _Inout_opt_ LPDWORD *pdwType, _Out_opt_ LPVOID *pvData, _Inout_ LPDWORD *pcbData, _In_ BOOL fIgnoreHKCU, _In_opt_ LPVOID *pvDefaultData, _In_opt_ DWORD dwDefaultDataSize) |
| tsssuuui   | LSTATUS | SHRegSetPath(_In_ HKEY hkey, _In_ LPCTSTR pszSubkey, _In_ LPCTSTR pszValue, _In_ LPCTSTR pszPath, DWORD dwFlags)                                                                                                                             |
| taaauiii   | LSTATUS | SHRegSetPathA(_In_ HKEY hkey, _In_ LPCSTR pszSubkey, _In_ LPCSTR pszValue, _In_ LPCSTR pszPath, DWORD dwFlags)                                                                                                                               |
| twwwuiii   | LSTATUS | SHRegSetPathW(_In_ HKEY hkey, _In_ LPCWSTR pszSubkey, _In_ LPCWSTR pszValue, _In_ LPCWSTR pszPath, DWORD dwFlags)                                                                                                                            |
| ssuituiiii | LSTATUS | SHRegSetUSValue(_In_ LPCTSTR pszSubKey, _In_ LPCTSTR pszValue, _In_ DWORD dwType, _In_opt_ LPVOID *pvData, _In_opt_ DWORD cbData, _In_opt_ DWORD dwFlags)                                                                                    |
| aauiiiiii  | LSTATUS | SHRegSetUSValueA(_In_ LPCSTR pszSubKey, _In_ LPCSTR pszValue, _In_ DWORD dwType, _In_opt_ LPVOID *pvData, _In_opt_ DWORD cbData, _In_opt_ DWORD dwFlags)                                                                                     |
| wwuituiiii | LSTATUS | SHRegSetUSValueW(_In_ LPCWSTR pszSubKey, _In_ LPCWSTR pszValue, _In_ DWORD dwType, _In_opt_ LPVOID *pvData, _In_opt_ DWORD cbData, _In_opt_ DWORD dwFlags)                                                                                   |
| tsuituiiii | LSTATUS | SHRegWriteUSValue(_In_ HUSKEY hUSKey, _In_ LPCTSTR pszValue, _In_ DWORD dwType, _In_ const void *pvData, _In_ DWORD cbData, _In_ DWORD dwFlags)                                                                                              |
| tauituiiii | LSTATUS | SHRegWriteUSValueA(_In_ HUSKEY hUSKey, _In_ LPCSTR pszValue, _In_ DWORD dwType, _In_ const void *pvData, _In_ DWORD cbData, _In_ DWORD dwFlags)                                                                                              |
| twuituiiii | LSTATUS | SHRegWriteUSValueW(_In_ HUSKEY hUSKey, _In_ LPCWSTR pszValue, _In_ DWORD dwType, _In_ const void *pvData, _In_ DWORD cbData, _In_ DWORD dwFlags)                                                                                             |
| i          | HRESULT | SHReleaseThreadRef(void)                                                                                                                                                                                                                     |
| uittt      | LRESULT | SHSendMessageBroadcast(_In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                                                                                               |

|            |         |                                                                                                                                                   |
|------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| uittt      | LRESULT | SHSendMessageBroadcastA(_In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                   |
| uittt      | LRESULT | SHSendMessageBroadcastW(_In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                   |
| ti         | HRESULT | SHSetThreadRef(_In_opt_ IUnknown *pUnk)                                                                                                           |
| tssuituiui | LSTATUS | SHSetValue(_In_ HKEY hkey, _In_opt_ LPCTSTR pszSubKey, _In_opt_ LPCTSTR pszValue, _In_ DWORD dwType, _In_opt_ LPCVOID pvData, _In_ DWORD cbData)  |
| taaituiui  | LSTATUS | SHSetValueA(_In_ HKEY hkey, _In_opt_ LPCSTR pszSubKey, _In_opt_ LPCSTR pszValue, _In_ DWORD dwType, _In_opt_ LPCVOID pvData, _In_ DWORD cbData)   |
| twwuituiui | LSTATUS | SHSetValueW(_In_ HKEY hkey, _In_opt_ LPCWSTR pszSubKey, _In_opt_ LPCWSTR pszValue, _In_ DWORD dwType, _In_opt_ LPCVOID pvData, _In_ DWORD cbData) |
| tii        | BOOL    | SHSkipFunction(_In_opt_ IBindCtx *pbc, _In_ const CLSID *pclsid)                                                                                  |
| sti        | HRESULT | SHStrDup(_In_ LPCTSTR pszSource, _Out_ LPTSTR *ppwsz)                                                                                             |
| ati        | HRESULT | SHStrDupA(_In_ LPCSTR pszSource, _Out_ LPSTR *ppwsz)                                                                                              |
| wti        | HRESULT | SHStrDupW(_In_ LPCWSTR pszSource, _Out_ LPWSTR *ppwsz)                                                                                            |
| ty         | TCHAR   | SHStripMnemonic(_Inout_ LPTSTR *pszMenu)                                                                                                          |
| ty         | TCHAR   | SHStripMnemonicA(_Inout_ LPSTR *pszMenu)                                                                                                          |
| ty         | TCHAR   | SHStripMnemonicW(_Inout_ LPWSTR *pszMenu)                                                                                                         |
| waii       | int     | SHUnicodeToAns(_In_ PCWSTR pwszSrc, _Out_ PSTR pszDst, int cchBuf)                                                                                |
| wtii       | int     | SHUnicodeToUnicode(_In_ PCWSTR pwzSrc, _Out_ PWSTR pwzDst, int cwchBuf)                                                                           |
| ti         | BOOL    | SHUnlockShared(_In_ void *pvData)                                                                                                                 |
| sss        | PTSTR   | StrCat(_Inout_ PTSTR psz1, _In_ PCTSTR psz2)                                                                                                      |
| ssis       | PTSTR   | StrCatBuff(_Inout_ PTSTR pszDest, _In_ PCTSTR pszSrc, int cchDestBuffSize)                                                                        |
| aaia       | PTSTR   | StrCatBuffA(_Inout_ PTSTR pszDest, _In_ PCTSTR pszSrc, int cchDestBuffSize)                                                                       |
| wwiw       | PTSTR   | StrCatBuffW(_Inout_ PTSTR pszDest, _In_ PCTSTR pszSrc, int cchDestBuffSize)                                                                       |
| tuiuiwui   | DWORD   | StrCatChainW(_Out_ PWSTR pszDst, DWORD cchDst, DWORD ichAt, _In_ PCWSTR pszSrc)                                                                   |
| www        | PTSTR   | StrCatW(_Inout_ PTSTR psz1, _In_ PCTSTR psz2)                                                                                                     |
| sys        | PTSTR   | StrChr(_In_ PTSTR pszStart, TCHAR wMatch)                                                                                                         |
| aya        | PTSTR   | StrChrA(_In_ PTSTR pszStart, TCHAR wMatch)                                                                                                        |
| sys        | PTSTR   | StrChrI(_In_ PTSTR pszStart, TCHAR wMatch)                                                                                                        |

|        |       |                                                                   |
|--------|-------|-------------------------------------------------------------------|
| aya    | PTSTR | StrChrIA(_In_ PTSTR pszStart, TCHAR wMatch)                       |
| wyw    | PTSTR | StrChrIW(_In_ PTSTR pszStart, TCHAR wMatch)                       |
| wuhuit | PWSTR | StrChrNIW(_In_ PCWSTR pszStart, WCHAR wMatch, UINT cchMax)        |
| tuhuit | PWSTR | StrChrNW(_In_ PWSTR pszStart, WCHAR wMatch, UINT cchMax)          |
| wyw    | PTSTR | StrChrW(_In_ PTSTR pszStart, TCHAR wMatch)                        |
| ssi    | int   | StrCmp(_In_ PCTSTR psz1, _In_ PCTSTR psz2)                        |
| ssi    | int   | StrCmpC(_Out_ LPCTSTR lpStr1, _Out_ LPCTSTR lpStr2)               |
| aa1    | int   | StrCmpCA(_Out_ LPCSTR lpStr1, _Out_ LPCSTR lpStr2)                |
| ww1    | int   | StrCmpCW(_Out_ LPCWSTR lpStr1, _Out_ LPCWSTR lpStr2)              |
| ssi    | int   | StrCmpI(_In_ PCTSTR psz1, _In_ PCTSTR psz2)                       |
| ssi    | int   | StrCmpIC(_In_ LPCTSTR lpStr1, _In_ LPCTSTR lpStr2)                |
| aa1    | int   | StrCmpICA(_In_ LPCSTR lpStr1, _In_ LPCSTR lpStr2)                 |
| ww1    | int   | StrCmpICW(_In_ LPCWSTR lpStr1, _In_ LPCWSTR lpStr2)               |
| ww1    | int   | StrCmpIW(_In_ PCTSTR psz1, _In_ PCTSTR psz2)                      |
| ww1    | int   | StrCmpLogicalW(_In_ PCWSTR psz1, _In_ PCWSTR psz2)                |
| ssii   | int   | StrCmpN(_In_ PCTSTR psz1, _In_ PCTSTR psz2, _In_ int nChar)       |
| aa11   | int   | StrCmpNA(_In_ PCTSTR psz1, _In_ PCTSTR psz2, _In_ int nChar)      |
| ssii   | int   | StrCmpNC(_In_ LPCTSTR pszStr1, _In_ LPCTSTR pszStr2, int nChar)   |
| aa11   | int   | StrCmpNCA(_In_ LPCSTR pszStr1, _In_ LPCSTR pszStr2, int nChar)    |
| wwii   | int   | StrCmpNCW(_In_ LPCWSTR pszStr1, _In_ LPCWSTR pszStr2, int nChar)  |
| ssii   | int   | StrCmpNI(_In_ PCTSTR psz1, _In_ PCTSTR psz2, _In_ int nChar)      |
| aa11   | int   | StrCmpNIA(_In_ PCTSTR psz1, _In_ PCTSTR psz2, _In_ int nChar)     |
| ssii   | int   | StrCmpNIC(_In_ LPCTSTR pszStr1, _In_ LPCTSTR pszStr2, int nChar)  |
| aa11   | int   | StrCmpNICA(_In_ LPCSTR pszStr1, _In_ LPCSTR pszStr2, int nChar)   |
| wwii   | int   | StrCmpNICW(_In_ LPCWSTR pszStr1, _In_ LPCWSTR pszStr2, int nChar) |
| wwii   | int   | StrCmpNIW(_In_ PCTSTR psz1, _In_ PCTSTR psz2, _In_ int nChar)     |
| wwii   | int   | StrCmpNW(_In_ PCTSTR psz1, _In_ PCTSTR psz2, _In_ int nChar)      |
| ww1    | int   | StrCmpW(_In_ PCTSTR psz1, _In_ PCTSTR psz2)                       |
| sss    | PTSTR | StrCpy(_Out_ PTSTR psz1, _In_ PCTSTR psz2)                        |
| ssis   | PTSTR | StrCpyN(_Out_ PTSTR pszDst, _In_ PCTSTR pszSrc, int cchMax)       |
| wwiw   | PTSTR | StrCpyNW(_Out_ PTSTR pszDst, _In_ PCTSTR pszSrc, int cchMax)      |
| www    | PTSTR | StrCpyW(_Out_ PTSTR psz1, _In_ PCTSTR psz2)                       |
| ssi    | int   | StrCSpt(_In_ PCTSTR pszStr, _In_ PCTSTR pszSet)                   |

|           |         |                                                                                           |
|-----------|---------|-------------------------------------------------------------------------------------------|
| aai       | int     | StrCSpnA(_In_ PCTSTR pszStr, _In_ PCTSTR pszSet)                                          |
| ssi       | int     | StrCSpnA(_In_ PCTSTR pszStr, _In_ PCTSTR pszSet)                                          |
| aai       | int     | StrCSpnA(_In_ PCTSTR pszStr, _In_ PCTSTR pszSet)                                          |
| wwi       | int     | StrCSpnW(_In_ PCTSTR pszStr, _In_ PCTSTR pszSet)                                          |
| wwi       | int     | StrCSpnW(_In_ PCTSTR pszStr, _In_ PCTSTR pszSet)                                          |
| ss        | PTSTR   | StrDup(PCTSTR pszSrch)                                                                    |
| aa        | PTSTR   | StrDupA(PCTSTR pszSrch)                                                                   |
| ww        | PTSTR   | StrDupW(PCTSTR pszSrch)                                                                   |
| i6auia    | PSTR    | StrFormatByteSize64(LONGLONG qdw, _Out_ PSTR pszBuf, UINT cchBuf)                         |
| i6auia    | PSTR    | StrFormatByteSize64A(LONGLONG qdw, _Out_ PSTR pszBuf, UINT cchBuf)                        |
| uiauia    | PSTR    | StrFormatByteSizeA(DWORD dw, _Out_ PSTR pszBuf, UINT cchBuf)                              |
| ui6uituii | HRESULT | StrFormatByteSizeE(ULONGLONG ull, SFBS_FLAGS flags, _Out_ PWSTR pszBuf, UINT cchBuf)      |
| i6tuit    | PWSTR   | StrFormatByteSizeW(LONGLONG qdw, _Out_ PWSTR pszBuf, UINT cchBuf)                         |
| i6suis    | PTSTR   | StrFormatKBSizA(LONGLONG qdw, _Out_ PTSTR pszBuf, UINT cchBuf)                            |
| i6auia    | PTSTR   | StrFormatKBSizA(LONGLONG qdw, _Out_ PTSTR pszBuf, UINT cchBuf)                            |
| i6wuiw    | PTSTR   | StrFormatKBSizW(LONGLONG qdw, _Out_ PTSTR pszBuf, UINT cchBuf)                            |
| suiuiii   | int     | StrFromTimeInterval(_Out_ PTSTR pszOut, UINT cchMax, DWORD dwTimeMS, int digits)          |
| auuiiii   | int     | StrFromTimeIntervalA(_Out_ PTSTR pszOut, UINT cchMax, DWORD dwTimeMS, int digits)         |
| wuiuiii   | int     | StrFromTimeIntervalW(_Out_ PTSTR pszOut, UINT cchMax, DWORD dwTimeMS, int digits)         |
| issii     | BOOL    | StrIsIntEqual(BOOL fCaseSens, _In_ PCTSTR pszString1, _In_ PCTSTR pszString2, int nChar)  |
| iaaia     | BOOL    | StrIsIntEqualA(BOOL fCaseSens, _In_ PCTSTR pszString1, _In_ PCTSTR pszString2, int nChar) |
| iwwii     | BOOL    | StrIsIntEqualW(BOOL fCaseSens, _In_ PCTSTR pszString1, _In_ PCTSTR pszString2, int nChar) |
| ssis      | PTSTR   | StrNCat(_Inout_ PTSTR psz1, PCTSTR psz2, int cchMax)                                      |
| aaia      | PTSTR   | StrNCatA(_Inout_ PTSTR psz1, PCTSTR psz2, int cchMax)                                     |
| wwiw      | PTSTR   | StrNCatW(_Inout_ PTSTR psz1, PCTSTR psz2, int cchMax)                                     |
| sss       | PTSTR   | StrPDel(_In_ PTSTR psz, _In_ PCTSTR pszSet)                                               |

|        |         |                                                                                                      |
|--------|---------|------------------------------------------------------------------------------------------------------|
| aaa    | PTSTR   | StrPBkA(_In_ PTSTR psz, _In_ PCTSTR pszSet)                                                          |
| www    | PTSTR   | StrPBkW(_In_ PTSTR psz, _In_ PCTSTR pszSet)                                                          |
| ssys   | PTSTR   | StrRChr(_In_ PTSTR pszStart, _In_opt_ PCTSTR pszEnd, TCHAR wMatch)                                   |
| aaya   | PTSTR   | StrRChrA(_In_ PTSTR pszStart, _In_opt_ PCTSTR pszEnd, TCHAR wMatch)                                  |
| ssys   | PTSTR   | StrRChr(_In_ PTSTR pszStart, _In_opt_ PCTSTR pszEnd, TCHAR wMatch)                                   |
| aaya   | PTSTR   | StrRChrA(_In_ PTSTR pszStart, _In_opt_ PCTSTR pszEnd, TCHAR wMatch)                                  |
| wwyw   | PTSTR   | StrRChrW(_In_ PTSTR pszStart, _In_opt_ PCTSTR pszEnd, TCHAR wMatch)                                  |
| wwyw   | PTSTR   | StrRChrW(_In_ PTSTR pszStart, _In_opt_ PCTSTR pszEnd, TCHAR wMatch)                                  |
| tti    | HRESULT | StrRetToBSTR(_Inout_ STRRET *pstr, _In_ PCUITEMID_CHILD pidl, _Out_ BSTR *pbstr)                     |
| ttsuii | HRESULT | StrRetToBuf(_Inout_ STRRET *pstr, _In_ PCUITEMID_CHILD pidl, _Out_ LPTSTR pszBuf, _In_ UINT cchBuf)  |
| ttauii | HRESULT | StrRetToBufA(_Inout_ STRRET *pstr, _In_ PCUITEMID_CHILD pidl, _Out_ LPSTR pszBuf, _In_ UINT cchBuf)  |
| ttwuii | HRESULT | StrRetToBufW(_Inout_ STRRET *pstr, _In_ PCUITEMID_CHILD pidl, _Out_ LPWSTR pszBuf, _In_ UINT cchBuf) |
| tti    | HRESULT | StrRetToStr(_Inout_ STRRET *pstr, _In_opt_ PCUITEMID_CHILD pidl, _Out_ LPTSTR *ppszName)             |
| tti    | HRESULT | StrRetToStrA(_Inout_ STRRET *pstr, _In_opt_ PCUITEMID_CHILD pidl, _Out_ LPSTR *ppszName)             |
| tti    | HRESULT | StrRetToStrW(_Inout_ STRRET *pstr, _In_opt_ PCUITEMID_CHILD pidl, _Out_ LPWSTR *ppszName)            |
| ssss   | PTSTR   | StrRSuf(_In_ PTSTR pszSource, _In_opt_ PCTSTR pszLast, _In_ PCTSTR pszSrch)                          |
| aaaa   | PTSTR   | StrRSuLA(_In_ PTSTR pszSource, _In_opt_ PCTSTR pszLast, _In_ PCTSTR pszSrch)                         |
| wwwwww | PTSTR   | StrRSuW(_In_ PTSTR pszSource, _In_opt_ PCTSTR pszLast, _In_ PCTSTR pszSrch)                          |
| ssi    | int     | StrSpn(_In_ PCTSTR psz, _In_ PCTSTR pszSet)                                                          |
| aaai   | int     | StrSpnA(_In_ PCTSTR psz, _In_ PCTSTR pszSet)                                                         |
| wwwi   | int     | StrSpnW(_In_ PCTSTR psz, _In_ PCTSTR pszSet)                                                         |
| sss    | PTSTR   | StrStr(_In_ PTSTR pszFirst, _In_ PCTSTR pszSrch)                                                     |
| aaa    | PTSTR   | StrStrA(_In_ PTSTR pszFirst, _In_ PCTSTR pszSrch)                                                    |
|        |         |                                                                                                      |

|        |         |                                                                                                                     |
|--------|---------|---------------------------------------------------------------------------------------------------------------------|
| sss    | PTSTR   | StrStr(_In_ PTSTR pszFirst, _In_ PCTSTR pszSrch)                                                                    |
| aaa    | PTSTR   | StrStrA(_In_ PTSTR pszFirst, _In_ PCTSTR pszSrch)                                                                   |
| www    | PTSTR   | StrStrW(_In_ PTSTR pszFirst, _In_ PCTSTR pszSrch)                                                                   |
| twuit  | PWSTR   | StrStrNW(_In_ PWSTR pszFirst, _In_ PCWSTR pszSrch, UINT cchMax)                                                     |
| twuit  | PWSTR   | StrStrNW(_In_ PWSTR pszFirst, _In_ PCWSTR pszSrch, UINT cchMax)                                                     |
| www    | PTSTR   | StrStrW(_In_ PTSTR pszFirst, _In_ PCTSTR pszSrch)                                                                   |
| si     | int     | StrToInt(_In_ PCTSTR pszSrc)                                                                                        |
| siti   | BOOL    | StrToInt64Ex(_In_ PCTSTR pszString, STIF_FLAGS dwFlags, _Out_ LONGLONG *pllRet)                                     |
| aiti   | BOOL    | StrToInt64ExA(_In_ PCTSTR pszString, STIF_FLAGS dwFlags, _Out_ LONGLONG *pllRet)                                    |
| witi   | BOOL    | StrToInt64ExW(_In_ PCTSTR pszString, STIF_FLAGS dwFlags, _Out_ LONGLONG *pllRet)                                    |
| ai     | int     | StrToIntA(_In_ PCTSTR pszSrc)                                                                                       |
| siti   | BOOL    | StrToIntEx(_In_ PCTSTR pszString, STIF_FLAGS dwFlags, _Out_ int *piRet)                                             |
| aiti   | BOOL    | StrToIntExA(_In_ PCTSTR pszString, STIF_FLAGS dwFlags, _Out_ int *piRet)                                            |
| witi   | BOOL    | StrToIntExW(_In_ PCTSTR pszString, STIF_FLAGS dwFlags, _Out_ int *piRet)                                            |
| wi     | int     | StrToIntW(_In_ PCTSTR pszSrc)                                                                                       |
| ssi    | BOOL    | StrTrim(_Inout_ PTSTR psz, _In_ PCTSTR pszTrimChars)                                                                |
| aai    | BOOL    | StrTrimA(_Inout_ PTSTR psz, _In_ PCTSTR pszTrimChars)                                                               |
| wwi    | BOOL    | StrTrimW(_Inout_ PTSTR psz, _In_ PCTSTR pszTrimChars)                                                               |
| sstuii | HRESULT | UrlApplyScheme(_In_ PCTSTR pszIn, _Out_ PTSTR pszOut, _Inout_ DWORD *pcchOut, DWORD dwFlags)                        |
| aatuii | HRESULT | UrlApplySchemeA(_In_ PCTSTR pszIn, _Out_ PTSTR pszOut, _Inout_ DWORD *pcchOut, DWORD dwFlags)                       |
| wwtuii | HRESULT | UrlApplySchemeW(_In_ PCTSTR pszIn, _Out_ PTSTR pszOut, _Inout_ DWORD *pcchOut, DWORD dwFlags)                       |
| sstuii | HRESULT | UrlCanonicalize(_In_ PCTSTR pszUrl, _Out_ PTSTR pszCanonicalized, _Inout_ DWORD *pcchCanonicalized, DWORD dwFlags)  |
| aatuii | HRESULT | UrlCanonicalizeA(_In_ PCTSTR pszUrl, _Out_ PTSTR pszCanonicalized, _Inout_ DWORD *pcchCanonicalized, DWORD dwFlags) |
| wwtuii | HRESULT | UrlCanonicalizeW(_In_ PCTSTR pszUrl, _Out_ PTSTR pszCanonicalized, _Inout_ DWORD *pcchCanonicalized, DWORD dwFlags) |

|           |         |                                                                                                                                    |
|-----------|---------|------------------------------------------------------------------------------------------------------------------------------------|
| ssstuii   | HRESULT | UrlCombine(_In_ PCTSTR pszBase, _In_ PCTSTR pszRelative, _Out_opt_ PTSTR pszCombined, _Inout_ DWORD *pcchCombined, DWORD dwFlags)  |
| aaatuii   | HRESULT | UrlCombineA(_In_ PCTSTR pszBase, _In_ PCTSTR pszRelative, _Out_opt_ PTSTR pszCombined, _Inout_ DWORD *pcchCombined, DWORD dwFlags) |
| wwwtuii   | HRESULT | UrlCombineW(_In_ PCTSTR pszBase, _In_ PCTSTR pszRelative, _Out_opt_ PTSTR pszCombined, _Inout_ DWORD *pcchCombined, DWORD dwFlags) |
| ssii      | int     | UrlCompare(_In_ PCTSTR psz1, _In_ PCTSTR psz2, BOOL fIgnoreSlash)                                                                  |
| aaaii     | int     | UrlCompareA(_In_ PCTSTR psz1, _In_ PCTSTR psz2, BOOL fIgnoreSlash)                                                                 |
| wwi       | int     | UrlCompareW(_In_ PCTSTR psz1, _In_ PCTSTR psz2, BOOL fIgnoreSlash)                                                                 |
| ssstuii   | HRESULT | UrlCreateFromPath(_In_ PCTSTR pszPath, _Out_ PTSTR pszUrl, _Inout_ DWORD *pcchUrl, DWORD dwFlags)                                  |
| aatuii    | HRESULT | UrlCreateFromPathA(_In_ PCTSTR pszPath, _Out_ PTSTR pszUrl, _Inout_ DWORD *pcchUrl, DWORD dwFlags)                                 |
| wwtuii    | HRESULT | UrlCreateFromPathW(_In_ PCTSTR pszPath, _Out_ PTSTR pszUrl, _Inout_ DWORD *pcchUrl, DWORD dwFlags)                                 |
| ssstuii   | HRESULT | UrlEscape(_In_ PCTSTR pszURL, _Out_ PTSTR pszEscaped, _Inout_ DWORD *pcchEscaped, DWORD dwFlags)                                   |
| aatuii    | HRESULT | UrlEscapeA(_In_ PCTSTR pszURL, _Out_ PTSTR pszEscaped, _Inout_ DWORD *pcchEscaped, DWORD dwFlags)                                  |
| wwtuii    | HRESULT | UrlEscapeW(_In_ PCTSTR pszURL, _Out_ PTSTR pszEscaped, _Inout_ DWORD *pcchEscaped, DWORD dwFlags)                                  |
| wtuii     | HRESULT | UrlFixupW(_In_ PCWSTR pcszUrl, _Out_ PWSTR pszTranslatedUrl, DWORD cchMax)                                                         |
| ss        | LPCTSTR | UrlGetLocation(_In_ PCTSTR pszURL)                                                                                                 |
| aa        | LPCSTR  | UrlGetLocationA(_In_ PCTSTR pszURL)                                                                                                |
| ww        | LPCWSTR | UrlGetLocationW(_In_ PCTSTR pszURL)                                                                                                |
| ssstuiiii | HRESULT | UrlGetPart(_In_ PCTSTR pszIn, _Out_ PTSTR pszOut, _Inout_ DWORD *pcchOut, DWORD dwPart, DWORD dwFlags)                             |
| aatuiiii  | HRESULT | UrlGetPartA(_In_ PCTSTR pszIn, _Out_ PTSTR pszOut, _Inout_ DWORD *pcchOut, DWORD dwPart, DWORD dwFlags)                            |
| wwtuiiii  | HRESULT | UrlGetPartW(_In_ PCTSTR pszIn, _Out_ PTSTR pszOut, _Inout_ DWORD *pcchOut, DWORD dwPart, DWORD dwFlags)                            |
| stuii     | HRESULT | UrlHash(_In_ PCTSTR pszURL, _Out_ BYTE *pbHash, DWORD cbHash)                                                                      |
| atuii     | HRESULT | UrlHashA(_In_ PCTSTR pszURL, _Out_ BYTE *pbHash, DWORD                                                                             |

|        |         |                                                                                                                   |
|--------|---------|-------------------------------------------------------------------------------------------------------------------|
|        |         | cbHash)                                                                                                           |
| wtuui  | HRESULT | UrlHashW(_In_ PCTSTR pszURL, _Out_ BYTE *pbHash, DWORD cbHash)                                                    |
| suii   | BOOL    | Urls(_In_ PCTSTR pszUrl, URLIS UriIs)                                                                             |
| auui   | BOOL    | UrlsA(_In_ PCTSTR pszUrl, URLIS UriIs)                                                                            |
| si     | BOOL    | UrlsNoHistory(_In_ PCTSTR pszURL)                                                                                 |
| ai     | BOOL    | UrlsNoHistoryA(_In_ PCTSTR pszURL)                                                                                |
| wi     | BOOL    | UrlsNoHistoryW(_In_ PCTSTR pszURL)                                                                                |
| si     | BOOL    | UrlsOpaque(_In_ PCTSTR pszURL)                                                                                    |
| ai     | BOOL    | UrlsOpaqueA(_In_ PCTSTR pszURL)                                                                                   |
| wi     | BOOL    | UrlsOpaqueW(_In_ PCTSTR pszURL)                                                                                   |
| wuui   | BOOL    | UrlsW(_In_ PCTSTR pszUrl, URLIS UriIs)                                                                            |
| sstuii | HRESULT | UrlUnescape(_Inout_ PTSTR pszURL, _Out_opt_ PTSTR pszUnescaped, _Inout_opt_ DWORD *pcchUnescaped, DWORD dwFlags)  |
| aatuii | HRESULT | UrlUnescapeA(_Inout_ PTSTR pszURL, _Out_opt_ PTSTR pszUnescaped, _Inout_opt_ DWORD *pcchUnescaped, DWORD dwFlags) |
| wwtuii | HRESULT | UrlUnescapeW(_Inout_ PTSTR pszURL, _Out_opt_ PTSTR pszUnescaped, _Inout_opt_ DWORD *pcchUnescaped, DWORD dwFlags) |
| ui     | UINT    | WhichPlatform(void)                                                                                               |
| sisti  | int     | wnsprintf(_Out_ PTSTR pszDest, _In_ int cchDest, _In_ PCTSTR pszFmt, _In_ ...)                                    |
| aiati  | int     | wnsprintfA(_Out_ PTSTR pszDest, _In_ int cchDest, _In_ PCTSTR pszFmt, _In_ ...)                                   |
| wiwti  | int     | wnsprintfW(_Out_ PTSTR pszDest, _In_ int cchDest, _In_ PCTSTR pszFmt, _In_ ...)                                   |
| sisti  | int     | wvnsprintf(_Out_ PTSTR pszDest, _In_ int cchDest, _In_ PCTSTR pszFmt, _In_ va_list arglist)                       |
| aiati  | int     | wvnsprintfA(_Out_ PTSTR pszDest, _In_ int cchDest, _In_ PCTSTR pszFmt, _In_ va_list arglist)                      |
| wiwti  | int     | wvnsprintfW(_Out_ PTSTR pszDest, _In_ int cchDest, _In_ PCTSTR pszFmt, _In_ va_list arglist)                      |

# Tapi32.dll

|             |      |                                                                                                                                                                              |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiauiui     | LONG | <a href="#">lineAccept</a> (HCALL hCall, LPCSTR lpszUserUserInfo, DWORD dwSize)                                                                                              |
| attui       | LONG | <a href="#">lineAddProvider</a> (LPCSTR lpszProviderFilename, HWND hwndOwner, LPDWORD lpdwPermanentProviderID)                                                               |
| attui       | LONG | <a href="#">lineAddProviderA</a> (LPCSTR lpszProviderFilename, HWND hwndOwner, LPDWORD lpdwPermanentProviderID)                                                              |
| attui       | LONG | <a href="#">lineAddProviderW</a> (LPCSTR lpszProviderFilename, HWND hwndOwner, LPDWORD lpdwPermanentProviderID)                                                              |
| uiuiui      | LONG | <a href="#">lineAddToConference</a> (HCALL hConfCall, HCALL hConsultCall)                                                                                                    |
| uiuiuituiui | LONG | <a href="#">lineAgentSpecific</a> (HLINE hLine, DWORD dwAddressID, DWORD dwAgentExtensionIDIndex, LPVOID lpParams, DWORD dwSize)                                             |
| uiauiui     | LONG | <a href="#">lineAnswer</a> (HCALL hCall, LPCSTR lpszUserUserInfo, DWORD dwSize)                                                                                              |
| uiauiui     | LONG | <a href="#">lineBlindTransfer</a> (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                 |
| uiauiui     | LONG | <a href="#">lineBlindTransferA</a> (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                |
| uiauiui     | LONG | <a href="#">lineBlindTransferW</a> (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                |
| uiui        | LONG | <a href="#">lineClose</a> (HLINE hLine)                                                                                                                                      |
| uituiuiui   | LONG | <a href="#">lineCompleteCall</a> (HCALL hCall, LPDWORD lpdwCompletionID, DWORD dwCompletionMode, DWORD dwMessageID)                                                          |
| uiuituiui   | LONG | <a href="#">lineCompleteTransfer</a> (HCALL hCall, HCALL hConsultCall, LPHCALL lphConfCall, DWORD dwTransferMode)                                                            |
| uitau       | LONG | <a href="#">lineConfigDialog</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass)                                                                                  |
| uitau       | LONG | <a href="#">lineConfigDialogA</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass)                                                                                 |
| uitatuisui  | LONG | <a href="#">lineConfigDialogEdit</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass, LPVOID const lpDeviceConfigIn, DWORD dwSize, LPVARSTRING lpDeviceConfigOut)  |
| uitatuiiai  | LONG | <a href="#">lineConfigDialogEditA</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass, LPVOID const lpDeviceConfigIn, DWORD dwSize, LPVARSTRING lpDeviceConfigOut) |
| uitatuiwui  | LONG | <a href="#">lineConfigDialogEditW</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass, LPVOID const lpDeviceConfigIn, DWORD dwSize, LPVARSTRING lpDeviceConfigOut) |
| uitau       | LONG | <a href="#">lineConfigDialogW</a> (DWORD dwDeviceID, HWND hwndOwner,                                                                                                         |

|               |      |                                                                                                                                                                                                  |
|---------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |      | LPCSTR lpszDeviceClass)                                                                                                                                                                          |
| tuiui         | LONG | lineConfigProvider (HWND hwndOwner, DWORD dwPermanentProviderID)                                                                                                                                 |
| uiwwtui       | LONG | lineCreateAgent (HLINE hLine, LPWSTR lpszAgentID, LPWSTR lpszAgentPIN, LPHAGENT lphAgent)                                                                                                        |
| uiaatui       | LONG | lineCreateAgentA (HLINE hLine, LPWSTR lpszAgentID, LPWSTR lpszAgentPIN, LPHAGENT lphAgent)                                                                                                       |
| uiuiwuittui   | LONG | lineCreateAgentSession (HLINE hLine, HAGENT hAgent, LPWSTR lpszAgentPIN, DWORD dwWorkingAddressID, LPGUID lpGroupID, LPHAGENTSESSION lphAgentSession)                                            |
| uiuiauittui   | LONG | lineCreateAgentSessionA (HLINE hLine, HAGENT hAgent, LPWSTR lpszAgentPIN, DWORD dwWorkingAddressID, LPGUID lpGroupID, LPHAGENTSESSION lphAgentSession)                                           |
| uiuiwuittui   | LONG | lineCreateAgentSessionW (HLINE hLine, HAGENT hAgent, LPWSTR lpszAgentPIN, DWORD dwWorkingAddressID, LPGUID lpGroupID, LPHAGENTSESSION lphAgentSession)                                           |
| uiwwtui       | LONG | lineCreateAgentW (HLINE hLine, LPWSTR lpszAgentID, LPWSTR lpszAgentPIN, LPHAGENT lphAgent)                                                                                                       |
| uiui          | LONG | lineDeallocateCall (HCALL hCall)                                                                                                                                                                 |
| uiuiuituiui   | LONG | lineDevSpecific (HLINE hLine, DWORD dwAddressID, HCALL hCall, LPVOID lpParams, DWORD dwSize)                                                                                                     |
| uiuituiui     | LONG | lineDevSpecificFeature (HLINE hLine, DWORD dwFeature, LPVOID lpParams, DWORD dwSize)                                                                                                             |
| uiauiui       | LONG | lineDial (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                                                              |
| uiauiui       | LONG | lineDialA (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                                                             |
| uiauiui       | LONG | lineDialW (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                                                             |
| uiauiui       | LONG | lineDrop (HCALL hCall, LPCSTR lpszUserUserInfo, DWORD dwSize)                                                                                                                                    |
| uiuiuituittui | LONG | lineForward (HLINE hLine, DWORD bAllAddresses, DWORD dwAddressID, LPLINEFORWARDLIST const lpForwardList, DWORD dwNumRingsNoAnswer, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)  |
| uiuiuituittui | LONG | lineForwardA (HLINE hLine, DWORD bAllAddresses, DWORD dwAddressID, LPLINEFORWARDLIST const lpForwardList, DWORD dwNumRingsNoAnswer, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams) |
| uiuiuituittui | LONG | lineForwardW (HLINE hLine, DWORD bAllAddresses, DWORD dwAddressID, LPLINEFORWARDLIST const lpForwardList, DWORD dwNumRingsNoAnswer, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams) |
|               |      | lineGatherDigits (HCALL hCall, DWORD dwDigitModes, LPSTR                                                                                                                                         |

|               |      |                                                                                                                                                                            |
|---------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiauiuiuiui | LONG | lpsDigits, DWORD dwNumDigits, LPCSTR lpszTerminationDigits, DWORD dwFirstDigitTimeout, DWORD dwInterDigitTimeout)                                                          |
| uiuiauiuiuiui | LONG | lineGatherDigitsA(HCALL hCall, DWORD dwDigitModes, LPSTR lpsDigits, DWORD dwNumDigits, LPCSTR lpszTerminationDigits, DWORD dwFirstDigitTimeout, DWORD dwInterDigitTimeout) |
| uiuiauiuiuiui | LONG | lineGatherDigitsW(HCALL hCall, DWORD dwDigitModes, LPSTR lpsDigits, DWORD dwNumDigits, LPCSTR lpszTerminationDigits, DWORD dwFirstDigitTimeout, DWORD dwInterDigitTimeout) |
| uiuiauiui     | LONG | lineGenerateDigits(HCALL hCall, DWORD dwDigitMode, LPCSTR lpszDigits, DWORD dwDuration)                                                                                    |
| uiuiauiui     | LONG | lineGenerateDigitsA(HCALL hCall, DWORD dwDigitMode, LPCSTR lpszDigits, DWORD dwDuration)                                                                                   |
| uiuiauiui     | LONG | lineGenerateDigitsW(HCALL hCall, DWORD dwDigitMode, LPCSTR lpszDigits, DWORD dwDuration)                                                                                   |
| uiuiuiuiui    | LONG | lineGenerateTone(HCALL hCall, DWORD dwToneMode, DWORD dwDuration, DWORD dwNumTones, LPLINEGENERATETONE const lpTones)                                                      |
| uiuiuiuiuiui  | LONG | lineGetAddressCaps(HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAPIVersion, DWORD dwExtVersion, LPLINEADDRESSCAPS lpAddressCaps)                        |
| uiuiuiuiuiui  | LONG | lineGetAddressCapsA(HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAPIVersion, DWORD dwExtVersion, LPLINEADDRESSCAPS lpAddressCaps)                       |
| uiuiuiuiuiui  | LONG | lineGetAddressCapsW(HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAPIVersion, DWORD dwExtVersion, LPLINEADDRESSCAPS lpAddressCaps)                       |
| uituiauiui    | LONG | lineGetAddressID(HLINE hLine, LPDWORD lpdwAddressID, DWORD dwAddressMode, LPCSTR lpszAddress, DWORD dwSize)                                                                |
| uituiauiui    | LONG | lineGetAddressIDA(HLINE hLine, LPDWORD lpdwAddressID, DWORD dwAddressMode, LPCSTR lpszAddress, DWORD dwSize)                                                               |
| uituiauiui    | LONG | lineGetAddressIDW(HLINE hLine, LPDWORD lpdwAddressID, DWORD dwAddressMode, LPCSTR lpszAddress, DWORD dwSize)                                                               |
| uiuitui       | LONG | lineGetAddressStatus(HLINE hLine, DWORD dwAddressID, LPLINEADDRESSSTATUS lpAddressStatus)                                                                                  |
| uiuitui       | LONG | lineGetAddressStatusA(HLINE hLine, DWORD dwAddressID, LPLINEADDRESSSTATUS lpAddressStatus)                                                                                 |
| uiuitui       | LONG | lineGetAddressStatusW(HLINE hLine, DWORD dwAddressID, LPLINEADDRESSSTATUS lpAddressStatus)                                                                                 |
| uiuitui       | LONG | lineGetAgentActivityList(HLINE hLine, DWORD dwAddressID, LPLINEAGENTACTIVITYLIST lpAgentActivityList)                                                                      |
| uiuitui       | LONG | lineGetAgentActivityListA(HLINE hLine, DWORD dwAddressID, LPLINEAGENTACTIVITYLIST lpAgentActivityList)                                                                     |
|               |      | lineGetAgentActivityListW(HLINE hLine, DWORD dwAddressID,                                                                                                                  |

|             |      |                                                                                                                                                                         |
|-------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuitui     | LONG | LPLINEAGENTACTIVITYLIST lpAgentActivityList)                                                                                                                            |
| uiuiuiuitui | LONG | lineGetAgentCaps(HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAppAPIVersion, LPLINEAGENTCAPS lpAgentCaps)                                            |
| uiuiuiuitui | LONG | lineGetAgentCapsA(HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAppAPIVersion, LPLINEAGENTCAPS lpAgentCaps)                                           |
| uiuiuiuitui | LONG | lineGetAgentCapsW(HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAppAPIVersion, LPLINEAGENTCAPS lpAgentCaps)                                           |
| uiuitui     | LONG | lineGetAgentGroupList(HLINE hLine, DWORD dwAddressID, LPLINEAGENTGROUPLIST lpAgentGroupList)                                                                            |
| uiuitui     | LONG | lineGetAgentGroupListA(HLINE hLine, DWORD dwAddressID, LPLINEAGENTGROUPLIST lpAgentGroupList)                                                                           |
| uiuitui     | LONG | lineGetAgentGroupListW(HLINE hLine, DWORD dwAddressID, LPLINEAGENTGROUPLIST lpAgentGroupList)                                                                           |
| uiuitui     | LONG | lineGetAgentInfo(HLINE hLine, HAGENT hAgent, LPLINEAGENTINFO lpAgentInfo)                                                                                               |
| uiuitui     | LONG | lineGetAgentSessionInfo(HLINE hLine, HAGENTSESSION hAgentSession, LPLINEAGENTSESSIONINFO lpAgentSessionInfo)                                                            |
| uiuitui     | LONG | lineGetAgentSessionList(HLINE hLine, HAGENT hAgent, LPLINEAGENTSESSIONLIST lpAgentSessionList)                                                                          |
| uiuitui     | LONG | lineGetAgentStatus(HLINE hLine, DWORD dwAddressID, LPLINEAGENTSTATUS lpAgentStatus)                                                                                     |
| uiuitui     | LONG | lineGetAgentStatusA(HLINE hLine, DWORD dwAddressID, LPLINEAGENTSTATUS lpAgentStatus)                                                                                    |
| uiuitui     | LONG | lineGetAgentStatusW(HLINE hLine, DWORD dwAddressID, LPLINEAGENTSTATUS lpAgentStatus)                                                                                    |
| auituistui  | LONG | lineGetAppPriority(LPCSTR lpszAppFilename, DWORD dwMediaMode, LPLINEEXTENSIONID lpExtensionID, DWORD dwRequestMode, LPVARSTRING lpExtensionName, LPDWORD lpdwPriority)  |
| auituiatui  | LONG | lineGetAppPriorityA(LPCSTR lpszAppFilename, DWORD dwMediaMode, LPLINEEXTENSIONID lpExtensionID, DWORD dwRequestMode, LPVARSTRING lpExtensionName, LPDWORD lpdwPriority) |
| auituiwtui  | LONG | lineGetAppPriorityW(LPCSTR lpszAppFilename, DWORD dwMediaMode, LPLINEEXTENSIONID lpExtensionID, DWORD dwRequestMode, LPVARSTRING lpExtensionName, LPDWORD lpdwPriority) |
| uitui       | LONG | lineGetCallInfo(HCALL hCall, LPLINECALLINFO lpCallInfo)                                                                                                                 |
| uitui       | LONG | lineGetCallInfoA(HCALL hCall, LPLINECALLINFO lpCallInfo)                                                                                                                |
| uitui       | LONG | lineGetCallInfoW(HCALL hCall, LPLINECALLINFO lpCallInfo)                                                                                                                |

|              |      |                                                                                                                                   |
|--------------|------|-----------------------------------------------------------------------------------------------------------------------------------|
| uitui        | LONG | <i>lineGetCallStatus</i> (HCALL hCall, LPLINECALLSTATUS lpCallStatus)                                                             |
| uitui        | LONG | <i>lineGetConfRelatedCalls</i> (HCALL hCall, LPLINECALLLIST lpCallList)                                                           |
| uiuitui      | LONG | <i>lineGetCountry</i> (DWORD dwCountryID, DWORD dwAPIVersion, LPLINECOUNTRYLIST lpLineCountryList)                                |
| uiuitui      | LONG | <i>lineGetCountryA</i> (DWORD dwCountryID, DWORD dwAPIVersion, LPLINECOUNTRYLIST lpLineCountryList)                               |
| uiuitui      | LONG | <i>lineGetCountryW</i> (DWORD dwCountryID, DWORD dwAPIVersion, LPLINECOUNTRYLIST lpLineCountryList)                               |
| uiuiuiuitui  | LONG | <i>lineGetDevCaps</i> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtVersion, LPLINEDEVCAPS lpLineDevCaps)  |
| uiuiuiuitui  | LONG | <i>lineGetDevCapsA</i> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtVersion, LPLINEDEVCAPS lpLineDevCaps) |
| uiuiuiuitui  | LONG | <i>lineGetDevCapsW</i> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtVersion, LPLINEDEVCAPS lpLineDevCaps) |
| uisaui       | LONG | <i>lineGetDevConfig</i> (DWORD dwDeviceID, LPVARSTRING lpDeviceConfig, LPCSTR lpszDeviceClass)                                    |
| uiaaui       | LONG | <i>lineGetDevConfigA</i> (DWORD dwDeviceID, LPVARSTRING lpDeviceConfig, LPCSTR lpszDeviceClass)                                   |
| uiwau        | LONG | <i>lineGetDevConfigW</i> (DWORD dwDeviceID, LPVARSTRING lpDeviceConfig, LPCSTR lpszDeviceClass)                                   |
| uitui        | LONG | <i>lineGetGroupList</i> (HLINE hLine, LPLINEAGENTGROUPLIST lpGroupList)                                                           |
| uitui        | LONG | <i>lineGetGroupListA</i> (HLINE hLine, LPLINEAGENTGROUPLIST lpGroupList)                                                          |
| uitui        | LONG | <i>lineGetGroupListW</i> (HLINE hLine, LPLINEAGENTGROUPLIST lpGroupList)                                                          |
| uiatui       | LONG | <i>lineGetIcon</i> (DWORD dwDeviceID, LPCSTR lpszDeviceClass, LPHICON lphIcon)                                                    |
| uiatui       | LONG | <i>lineGetIconA</i> (DWORD dwDeviceID, LPCSTR lpszDeviceClass, LPHICON lphIcon)                                                   |
| uiatui       | LONG | <i>lineGetIconW</i> (DWORD dwDeviceID, LPCSTR lpszDeviceClass, LPHICON lphIcon)                                                   |
| uiuiuiuisaui | LONG | <i>lineGetID</i> (HLINE hLine, DWORD dwAddressID, HCALL hCall, DWORD dwSelect, LPVARSTRING lpDeviceID, LPCSTR lpszDeviceClass)    |
| uiuiuiuiaaui | LONG | <i>lineGetIDA</i> (HLINE hLine, DWORD dwAddressID, HCALL hCall, DWORD dwSelect, LPVARSTRING lpDeviceID, LPCSTR lpszDeviceClass)   |

|             |      |                                                                                                                                                |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiuiuiwau | LONG | <a href="#">lineGetIDW</a> (HLINE hLine, DWORD dwAddressID, HCALL hCall, DWORD dwSelect, LPVARSTRING lpDeviceID, LPCSTR lpzDeviceClass)        |
| uitui       | LONG | <a href="#">lineGetLineDevStatus</a> (HLINE hLine, LPLINEDEVSTATUS lpLineDevStatus)                                                            |
| uitui       | LONG | <a href="#">lineGetLineDevStatusA</a> (HLINE hLine, LPLINEDEVSTATUS lpLineDevStatus)                                                           |
| uitui       | LONG | <a href="#">lineGetLineDevStatusW</a> (HLINE hLine, LPLINEDEVSTATUS lpLineDevStatus)                                                           |
| uituiui     | LONG | <a href="#">lineGetMessage</a> (HLINEAPP hLineApp, LPLINEMESSAGE lpMessage, DWORD dwTimeout)                                                   |
| uiuiuitui   | LONG | <a href="#">lineGetNewCalls</a> (HLINE hLine, DWORD dwAddressID, DWORD dwSelect, LPLINECALLLIST lpCallList)                                    |
| uiuitui     | LONG | <a href="#">lineGetNumRings</a> (HLINE hLine, DWORD dwAddressID, LPDWORD lpdwNumRings)                                                         |
| uitui       | LONG | <a href="#">lineGetProviderList</a> (DWORD dwAPIVersion, LPLINEPROVIDERLIST lpProviderList)                                                    |
| uitui       | LONG | <a href="#">lineGetProviderListA</a> (DWORD dwAPIVersion, LPLINEPROVIDERLIST lpProviderList)                                                   |
| uitui       | LONG | <a href="#">lineGetProviderListW</a> (DWORD dwAPIVersion, LPLINEPROVIDERLIST lpProviderList)                                                   |
| uiuiuitui   | LONG | <a href="#">lineGetProxyStatus</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAppAPIVersion, LPLINEPROXYREQUESTLIST lpLineProxyRequestList) |
| uiuitui     | LONG | <a href="#">lineGetQueueInfo</a> (HLINE hLine, DWORD dwQueueID, LPLINEQUEUEINFO lpLineQueueInfo)                                               |
| uittui      | LONG | <a href="#">lineGetQueueList</a> (HLINE hLine, LPGUID lpGroupID, LPLINEQUEUELIST lpQueueList)                                                  |
| uittui      | LONG | <a href="#">lineGetQueueListA</a> (HLINE hLine, LPGUID lpGroupID, LPLINEQUEUELIST lpQueueList)                                                 |
| uittui      | LONG | <a href="#">lineGetQueueListW</a> (HLINE hLine, LPGUID lpGroupID, LPLINEQUEUELIST lpQueueList)                                                 |
| uiuitui     | LONG | <a href="#">lineGetRequest</a> (HLINEAPP hLineApp, DWORD dwRequestMode, LPVOID lpRequestBuffer)                                                |
| uiuitui     | LONG | <a href="#">lineGetRequestA</a> (HLINEAPP hLineApp, DWORD dwRequestMode, LPVOID lpRequestBuffer)                                               |
| uiuitui     | LONG | <a href="#">lineGetRequestW</a> (HLINEAPP hLineApp, DWORD dwRequestMode, LPVOID lpRequestBuffer)                                               |
| uittui      | LONG | <a href="#">lineGetStatusMessages</a> (HLINE hLine, LPDWORD lpdwLineStates, LPDWORD lpdwAddressStates)                                         |
| uiuitui     | LONG | <a href="#">lineGetTranslateCaps</a> (HLINEAPP hLineApp, DWORD dwAPIVersion, LPLINETRANSLATECAPS lpTranslateCaps)                              |

|             |      |                                                                                                                                                                                                                                       |
|-------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuitui     | LONG | <a href="#">lineGetTranslateCapsA</a> (HLINEAPP hLineApp, DWORD dwAPIVersion, LPLINETRANSLATECAPS lpTranslateCaps)                                                                                                                    |
| uiuitui     | LONG | <a href="#">lineGetTranslateCapsW</a> (HLINEAPP hLineApp, DWORD dwAPIVersion, LPLINETRANSLATECAPS lpTranslateCaps)                                                                                                                    |
| uiauiui     | LONG | <a href="#">lineHandoff</a> (HCALL hCall, LPCSTR lpszFileName, DWORD dwMediaMode)                                                                                                                                                     |
| uiauiui     | LONG | <a href="#">lineHandoffA</a> (HCALL hCall, LPCSTR lpszFileName, DWORD dwMediaMode)                                                                                                                                                    |
| uiauiui     | LONG | <a href="#">lineHandoffW</a> (HCALL hCall, LPCSTR lpszFileName, DWORD dwMediaMode)                                                                                                                                                    |
| uiui        | LONG | <a href="#">lineHold</a> (HCALL hCall)                                                                                                                                                                                                |
| tttatu      | LONG | <a href="#">lineInitialize</a> (LPHLINEAPP lphLineApp, HINSTANCE hInstance, LINECALLBACK lpfncallback, LPCSTR lpszAppName, LPDWORD lpdwNumDevs)                                                                                       |
| tttattui    | LONG | <a href="#">lineInitializeEx</a> (LPHLINEAPP lphLineApp, HINSTANCE hInstance, LINECALLBACK lpfncallback, LPCSTR lpszFriendlyAppName, LPDWORD lpdwNumDevs, LPDWORD lpdwAPIVersion, LPLINEINITIALIZEEXPARAMS lpLineInitializeExParams)  |
| tttattui    | LONG | <a href="#">lineInitializeExA</a> (LPHLINEAPP lphLineApp, HINSTANCE hInstance, LINECALLBACK lpfncallback, LPCSTR lpszFriendlyAppName, LPDWORD lpdwNumDevs, LPDWORD lpdwAPIVersion, LPLINEINITIALIZEEXPARAMS lpLineInitializeExParams) |
| tttattui    | LONG | <a href="#">lineInitializeExW</a> (LPHLINEAPP lphLineApp, HINSTANCE hInstance, LINECALLBACK lpfncallback, LPCSTR lpszFriendlyAppName, LPDWORD lpdwNumDevs, LPDWORD lpdwAPIVersion, LPLINEINITIALIZEEXPARAMS lpLineInitializeExParams) |
| uitauitui   | LONG | <a href="#">lineMakeCall</a> (HLINE hLine, LPHCALL lphCall, LPCSTR lpszDestAddress, DWORD dwCountryCode, LPLINECALLPARAMS const lpCallParams)                                                                                         |
| uitauitui   | LONG | <a href="#">lineMakeCallA</a> (HLINE hLine, LPHCALL lphCall, LPCSTR lpszDestAddress, DWORD dwCountryCode, LPLINECALLPARAMS const lpCallParams)                                                                                        |
| uitauitui   | LONG | <a href="#">lineMakeCallW</a> (HLINE hLine, LPHCALL lphCall, LPCSTR lpszDestAddress, DWORD dwCountryCode, LPLINECALLPARAMS const lpCallParams)                                                                                        |
| uiuiui      | LONG | <a href="#">lineMonitorDigits</a> (HCALL hCall, DWORD dwDigitModes)                                                                                                                                                                   |
| uiuiui      | LONG | <a href="#">lineMonitorMedia</a> (HCALL hCall, DWORD dwMediaModes)                                                                                                                                                                    |
| uituiui     | LONG | <a href="#">lineMonitorTones</a> (HCALL hCall, LPLINEMONITORTONE const lpToneList, DWORD dwNumEntries)                                                                                                                                |
| uiuiuiittui | LONG | <a href="#">lineNegotiateAPIVersion</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPILowVersion, DWORD dwAPIHighVersion, LPDWORD lpdwAPIVersion, LPLINEEXTENSIONID lpExtensionID)                                                 |

|                   |      |                                                                                                                                                                                                                            |
|-------------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiuiuiuitui     | LONG | <i>lineNegotiateExtVersion</i> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtLowVersion, DWORD dwExtHighVersion, LPDWORD lpdwExtVersion)                                                            |
| uiuituiuituiuitui | LONG | <i>lineOpen</i> (HLINEAPP hLineApp, DWORD dwDeviceID, LPHLINE lphLine, DWORD dwAPIVersion, DWORD dwExtVersion, DWORD_PTR dwCallbackInstance, DWORD dwPrivileges, DWORD dwMediaModes, LPLINECALLPARAMS const lpCallParams)  |
| uiuituiuituiuitui | LONG | <i>lineOpenA</i> (HLINEAPP hLineApp, DWORD dwDeviceID, LPHLINE lphLine, DWORD dwAPIVersion, DWORD dwExtVersion, DWORD_PTR dwCallbackInstance, DWORD dwPrivileges, DWORD dwMediaModes, LPLINECALLPARAMS const lpCallParams) |
| uiuituiuituiuitui | LONG | <i>lineOpenW</i> (HLINEAPP hLineApp, DWORD dwDeviceID, LPHLINE lphLine, DWORD dwAPIVersion, DWORD dwExtVersion, DWORD_PTR dwCallbackInstance, DWORD dwPrivileges, DWORD dwMediaModes, LPLINECALLPARAMS const lpCallParams) |
| uiuiasui          | LONG | <i>linePark</i> (HCALL hCall, DWORD dwParkMode, LPCSTR lpszDirAddress, LPVARSTRING lpNonDirAddress)                                                                                                                        |
| uiuiaaui          | LONG | <i>lineParkA</i> (HCALL hCall, DWORD dwParkMode, LPCSTR lpszDirAddress, LPVARSTRING lpNonDirAddress)                                                                                                                       |
| uiuiawui          | LONG | <i>lineParkW</i> (HCALL hCall, DWORD dwParkMode, LPCSTR lpszDirAddress, LPVARSTRING lpNonDirAddress)                                                                                                                       |
| uiuitaui          | LONG | <i>linePickup</i> (HLINE hLine, DWORD dwAddressID, LPHCALL lphCall, LPCSTR lpszDestAddress, LPCSTR lpszGroupID)                                                                                                            |
| uiuitaui          | LONG | <i>linePickupA</i> (HLINE hLine, DWORD dwAddressID, LPHCALL lphCall, LPCSTR lpszDestAddress, LPCSTR lpszGroupID)                                                                                                           |
| uiuitaui          | LONG | <i>linePickupW</i> (HLINE hLine, DWORD dwAddressID, LPHCALL lphCall, LPCSTR lpszDestAddress, LPCSTR lpszGroupID)                                                                                                           |
| uittui            | LONG | <i>linePrepareAddToConference</i> (HCALL hConfCall, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)                                                                                                           |
| uittui            | LONG | <i>linePrepareAddToConferenceA</i> (HCALL hConfCall, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)                                                                                                          |
| uittui            | LONG | <i>linePrepareAddToConferenceW</i> (HCALL hConfCall, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)                                                                                                          |
| uiuiuiuiuiui      | LONG | <i>lineProxyMessage</i> (HLINE hLine, HCALL hCall, DWORD dwMsg, DWORD dwParam1, DWORD dwParam2, DWORD dwParam3)                                                                                                            |
| uituiui           | LONG | <i>lineProxyResponse</i> (HLINE hLine, LPLINEPROXYREQUEST lpProxyRequest, DWORD dwResult)                                                                                                                                  |
| uiauiui           | LONG | <i>lineRedirect</i> (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                                                                             |
| uiauiui           | LONG | <i>lineRedirectA</i> (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                                                                            |
| uiauiui           | LONG | <i>lineRedirectW</i> (HCALL hCall, LPCSTR lpszDestAddress, DWORD dwCountryCode)                                                                                                                                            |

|             |      |                                                                                                                                                                  |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |      | dwCountryCode)                                                                                                                                                   |
| uiuiuiuiui  | LONG | lineRegisterRequestRecipient(HLINEAPP hLineApp, DWORD dwRegistrationInstance, DWORD dwRequestMode, DWORD bEnable)                                                |
| uiui        | LONG | lineReleaseUserUserInfo(HCALL hCall)                                                                                                                             |
| uiui        | LONG | lineRemoveFromConference(HCALL hCall)                                                                                                                            |
| uitui       | LONG | lineRemoveProvider(DWORD dwPermanentProviderID, HWND hwndOwner)                                                                                                  |
| uiui        | LONG | lineSecureCall(HCALL hCall)                                                                                                                                      |
| uiauiui     | LONG | lineSendUserUserInfo(HCALL hCall, LPCSTR lpszUserUserInfo, DWORD dwSize)                                                                                         |
| uiuiuiui    | LONG | lineSetAgentActivity(HLINE hLine, DWORD dwAddressID, DWORD dwActivityID)                                                                                         |
| uiuitui     | LONG | lineSetAgentGroup(HLINE hLine, DWORD dwAddressID, LPLINEAGENTGROUPLIST lpAgentGroupList)                                                                         |
| uiuiuiui    | LONG | lineSetAgentMeasurementPeriod(HLINE hLine, HAGENT hAgent, DWORD dwMeasurementPeriod)                                                                             |
| uiuiuiuiui  | LONG | lineSetAgentSessionState(HLINE hLine, HAGENTSESSION hAgentSession, DWORD dwAgentSessionState, DWORD dwNextAgentSessionState)                                     |
| uiuiuiuiui  | LONG | lineSetAgentState(HLINE hLine, DWORD dwAddressID, DWORD dwAgentState, DWORD dwNextAgentState)                                                                    |
| uiuiuiuiui  | LONG | lineSetAgentStateEx(HLINE hLine, HAGENT hAgent, DWORD dwAgentState, DWORD dwNextAgentState)                                                                      |
| aituiauiui  | LONG | lineSetAppPriority(LPCSTR lpszAppFilename, DWORD dwMediaMode, LPLINEEXTENSIONID lpExtensionID, DWORD dwRequestMode, LPCSTR lpszExtensionName, DWORD dwPriority)  |
| aituiauiui  | LONG | lineSetAppPriorityA(LPCSTR lpszAppFilename, DWORD dwMediaMode, LPLINEEXTENSIONID lpExtensionID, DWORD dwRequestMode, LPCSTR lpszExtensionName, DWORD dwPriority) |
| aituiauiui  | LONG | lineSetAppPriorityW(LPCSTR lpszAppFilename, DWORD dwMediaMode, LPLINEEXTENSIONID lpExtensionID, DWORD dwRequestMode, LPCSTR lpszExtensionName, DWORD dwPriority) |
| uiuiui      | LONG | lineSetAppSpecific(HCALL hCall, DWORD dwAppSpecific)                                                                                                             |
| uituiui     | LONG | lineSetCallData(HCALL hCall, LPVOID lpCallData, DWORD dwSize)                                                                                                    |
| uiuiuiuitui | LONG | lineSetCallParams(HCALL hCall, DWORD dwBearerMode, DWORD dwMinRate, DWORD dwMaxRate, LPLINEDIALPARAMS const lpDialParams)                                        |
| uiuiui      | LONG | lineSetCallPrivilege(HCALL hCall, DWORD dwCallPrivilege)                                                                                                         |
| uituituiui  | LONG | lineSetCallQualityOfService(HCALL hCall, LPVOID lpSendingFlowspec, DWORD dwSendingFlowspecSize, LPVOID lpReceivingFlowspec, DWORD dwReceivingFlowspecSize)       |
|             |      |                                                                                                                                                                  |

|                      |      |                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuiui               | LONG | <a href="#">lineSetCallTreatment</a> (HCALL hCall, DWORD dwTreatment)                                                                                                                                                                                                                                                                                                                      |
| uiuiui               | LONG | <a href="#">lineSetCurrentLocation</a> (HLINEAPP hLineApp, DWORD dwLocation)                                                                                                                                                                                                                                                                                                               |
| uituiaui             | LONG | <a href="#">lineSetDevConfig</a> (DWORD dwDeviceID, LPVOID const lpDeviceConfig, DWORD dwSize, LPCSTR lpszDeviceClass)                                                                                                                                                                                                                                                                     |
| uituiaui             | LONG | <a href="#">lineSetDevConfigA</a> (DWORD dwDeviceID, LPVOID const lpDeviceConfig, DWORD dwSize, LPCSTR lpszDeviceClass)                                                                                                                                                                                                                                                                    |
| uituiaui             | LONG | <a href="#">lineSetDevConfigW</a> (DWORD dwDeviceID, LPVOID const lpDeviceConfig, DWORD dwSize, LPCSTR lpszDeviceClass)                                                                                                                                                                                                                                                                    |
| uiuiuiui             | LONG | <a href="#">lineSetLineDevStatus</a> (HLINE hLine, DWORD dwStatusToChange, DWORD fStatus)                                                                                                                                                                                                                                                                                                  |
| uiuiuiuituituituitui | LONG | <a href="#">lineSetMediaControl</a> (HLINE hLine, DWORD dwAddressID, HCALL hCall, DWORD dwSelect, LPLINEMEDIACONTROLDIGIT const lpDigitList, DWORD dwDigitNumEntries, LPLINEMEDIACONTROLMEDIA const lpMediaList, DWORD dwMediaNumEntries, LPLINEMEDIACONTROLTONE const lpToneList, DWORD dwToneNumEntries, LPLINEMEDIACONTROLCALLSTATE const lpCallStateList, DWORD dwCallStateNumEntries) |
| uiuiui               | LONG | <a href="#">lineSetMediaMode</a> (HCALL hCall, DWORD dwMediaModes)                                                                                                                                                                                                                                                                                                                         |
| uiuiuiui             | LONG | <a href="#">lineSetNumRings</a> (HLINE hLine, DWORD dwAddressID, DWORD dwNumRings)                                                                                                                                                                                                                                                                                                         |
| uiuiuiui             | LONG | <a href="#">lineSetQueueMeasurementPeriod</a> (HLINE hLine, DWORD dwQueueID, DWORD dwMeasurementPeriod)                                                                                                                                                                                                                                                                                    |
| uiuiuiui             | LONG | <a href="#">lineSetStatusMessages</a> (HLINE hLine, DWORD dwLineStates, DWORD dwAddressStates)                                                                                                                                                                                                                                                                                             |
| uiuiuiuiuiuiuiui     | LONG | <a href="#">lineSetTerminal</a> (HLINE hLine, DWORD dwAddressID, HCALL hCall, DWORD dwSelect, DWORD dwTerminalModes, DWORD dwTerminalID, DWORD bEnable)                                                                                                                                                                                                                                    |
| uiuiauiui            | LONG | <a href="#">lineSetTollList</a> (HLINEAPP hLineApp, DWORD dwDeviceID, LPCSTR lpszAddressIn, DWORD dwTollListOption)                                                                                                                                                                                                                                                                        |
| uiuiauiui            | LONG | <a href="#">lineSetTollListA</a> (HLINEAPP hLineApp, DWORD dwDeviceID, LPCSTR lpszAddressIn, DWORD dwTollListOption)                                                                                                                                                                                                                                                                       |
| uiuiauiui            | LONG | <a href="#">lineSetTollListW</a> (HLINEAPP hLineApp, DWORD dwDeviceID, LPCSTR lpszAddressIn, DWORD dwTollListOption)                                                                                                                                                                                                                                                                       |
| uiuittuitui          | LONG | <a href="#">lineSetupConference</a> (HCALL hCall, HLINE hLine, LPHCALL lphConfCall, LPHCALL lphConsultCall, DWORD dwNumParties, LPLINECALLPARAMS const lpCallParams)                                                                                                                                                                                                                       |
| uiuittuitui          | LONG | <a href="#">lineSetupConferenceA</a> (HCALL hCall, HLINE hLine, LPHCALL lphConfCall, LPHCALL lphConsultCall, DWORD dwNumParties, LPLINECALLPARAMS const lpCallParams)                                                                                                                                                                                                                      |
| uiuittuitui          | LONG | <a href="#">lineSetupConferenceW</a> (HCALL hCall, HLINE hLine, LPHCALL lphConfCall, LPHCALL lphConsultCall, DWORD dwNumParties, LPLINECALLPARAMS const lpCallParams)                                                                                                                                                                                                                      |

|                |      |                                                                                                                                                                                                        |
|----------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uittui         | LONG | <a href="#">lineSetupTransfer</a> (HCALL hCall, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)                                                                                           |
| uittui         | LONG | <a href="#">lineSetupTransferA</a> (HCALL hCall, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)                                                                                          |
| uittui         | LONG | <a href="#">lineSetupTransferW</a> (HCALL hCall, LPHCALL lphConsultCall, LPLINECALLPARAMS const lpCallParams)                                                                                          |
| uiui           | LONG | <a href="#">lineShutdown</a> (HLINEAPP hLineApp)                                                                                                                                                       |
| uiuiui         | LONG | <a href="#">lineSwapHold</a> (HCALL hActiveCall, HCALL hHeldCall)                                                                                                                                      |
| uiuiuiauiuitui | LONG | <a href="#">lineTranslateAddress</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, LPCSTR lpszAddressIn, DWORD dwCard, DWORD dwTranslateOptions, LPLINETRANSLATEOUTPUT lpTranslateOutput)  |
| uiuiuiauiuitui | LONG | <a href="#">lineTranslateAddressA</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, LPCSTR lpszAddressIn, DWORD dwCard, DWORD dwTranslateOptions, LPLINETRANSLATEOUTPUT lpTranslateOutput) |
| uiuiuiauiuitui | LONG | <a href="#">lineTranslateAddressW</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, LPCSTR lpszAddressIn, DWORD dwCard, DWORD dwTranslateOptions, LPLINETRANSLATEOUTPUT lpTranslateOutput) |
| uiuiuitai      | LONG | <a href="#">lineTranslateDialog</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, HWND hwndOwner, LPCSTR lpszAddressIn)                                                                    |
| uiuiuitai      | LONG | <a href="#">lineTranslateDialogA</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, HWND hwndOwner, LPCSTR lpszAddressIn)                                                                   |
| uiuiuitai      | LONG | <a href="#">lineTranslateDialogW</a> (HLINEAPP hLineApp, DWORD dwDeviceID, DWORD dwAPIVersion, HWND hwndOwner, LPCSTR lpszAddressIn)                                                                   |
| uiuiui         | LONG | <a href="#">lineUncompleteCall</a> (HLINE hLine, DWORD dwCompletionID)                                                                                                                                 |
| uiui           | LONG | <a href="#">lineUnhold</a> (HCALL hCall)                                                                                                                                                               |
| uiuitai        | LONG | <a href="#">lineUnpark</a> (HLINE hLine, DWORD dwAddressID, LPHCALL lphCall, LPCSTR lpszDestAddress)                                                                                                   |
| uiuitai        | LONG | <a href="#">lineUnparkA</a> (HLINE hLine, DWORD dwAddressID, LPHCALL lphCall, LPCSTR lpszDestAddress)                                                                                                  |
| uiuitai        | LONG | <a href="#">lineUnparkW</a> (HLINE hLine, DWORD dwAddressID, LPHCALL lphCall, LPCSTR lpszDestAddress)                                                                                                  |
| uiui           | LONG | <a href="#">phoneClose</a> (HPHONE hPhone)                                                                                                                                                             |
| uitai          | LONG | <a href="#">phoneConfigDialog</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass)                                                                                                           |
| uitai          | LONG | <a href="#">phoneConfigDialogA</a> (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass)                                                                                                          |

|             |      |                                                                                                                           |
|-------------|------|---------------------------------------------------------------------------------------------------------------------------|
| uitai       | LONG | phoneConfigDialogW (DWORD dwDeviceID, HWND hwndOwner, LPCSTR lpszDeviceClass)                                             |
| uituiui     | LONG | phoneDevSpecific (HPHONE hPhone, LPVOID lpParams, DWORD dwSize)                                                           |
| uiuitui     | LONG | phoneGetButtonInfo (HPHONE hPhone, DWORD dwButtonLampID, LPPHONEBUTTONINFO lpButtonInfo)                                  |
| uiuitui     | LONG | phoneGetButtonInfoA (HPHONE hPhone, DWORD dwButtonLampID, LPPHONEBUTTONINFO lpButtonInfo)                                 |
| uiuitui     | LONG | phoneGetButtonInfoW (HPHONE hPhone, DWORD dwButtonLampID, LPPHONEBUTTONINFO lpButtonInfo)                                 |
| uiuituiui   | LONG | phoneGetData (HPHONE hPhone, DWORD dwDataID, LPVOID lpData, DWORD dwSize)                                                 |
| uiuiuiuitui | LONG | phoneGetDevCaps (HPHONEAPP hPhoneApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtVersion, LPPHONECAPS lpPhoneCaps)  |
| uiuiuiuitui | LONG | phoneGetDevCapsA (HPHONEAPP hPhoneApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtVersion, LPPHONECAPS lpPhoneCaps) |
| uiuiuiuitui | LONG | phoneGetDevCapsW (HPHONEAPP hPhoneApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtVersion, LPPHONECAPS lpPhoneCaps) |
| uisui       | LONG | phoneGetDisplay (HPHONE hPhone, LPVARSTRING lpDisplay)                                                                    |
| uiuitui     | LONG | phoneGetGain (HPHONE hPhone, DWORD dwHookSwitchDev, LPDWORD lpdwGain)                                                     |
| uitui       | LONG | phoneGetHookSwitch (HPHONE hPhone, LPDWORD lpdwHookSwitchDevs)                                                            |
| uiatui      | LONG | phoneGetIcon (DWORD dwDeviceID, LPCSTR lpszDeviceClass, LPHICON lphIcon)                                                  |
| uiatui      | LONG | phoneGetIconA (DWORD dwDeviceID, LPCSTR lpszDeviceClass, LPHICON lphIcon)                                                 |
| uiatui      | LONG | phoneGetIconW (DWORD dwDeviceID, LPCSTR lpszDeviceClass, LPHICON lphIcon)                                                 |
| uisai       | LONG | phoneGetID (HPHONE hPhone, LPVARSTRING lpDeviceID, LPCSTR lpszDeviceClass)                                                |
| uiaai       | LONG | phoneGetIDA (HPHONE hPhone, LPVARSTRING lpDeviceID, LPCSTR lpszDeviceClass)                                               |
| uiwai       | LONG | phoneGetIDW (HPHONE hPhone, LPVARSTRING lpDeviceID, LPCSTR lpszDeviceClass)                                               |
| uiuitui     | LONG | phoneGetLamp (HPHONE hPhone, DWORD dwButtonLampID, LPDWORD lpdwLampMode)                                                  |
| uituiui     | LONG | phoneGetMessage (HPHONEAPP hPhoneApp, LPPHONEMESSAGE lpMessage, DWORD dwTimeout)                                          |

|               |      |                                                                                                                                                                                                                            |
|---------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uittui        | LONG | phoneGetRing(HPHONE hPhone, LPDWORD lpdwRingMode, LPDWORD lpdwVolume)                                                                                                                                                      |
| uitui         | LONG | phoneGetStatus(HPHONE hPhone, LPPHONESTATUS lpPhoneStatus)                                                                                                                                                                 |
| uitui         | LONG | phoneGetStatusA(HPHONE hPhone, LPPHONESTATUS lpPhoneStatus)                                                                                                                                                                |
| uitttui       | LONG | phoneGetStatusMessages(HPHONE hPhone, LPDWORD lpdwPhoneStates, LPDWORD lpdwButtonModes, LPDWORD lpdwButtonStates)                                                                                                          |
| uitui         | LONG | phoneGetStatusW(HPHONE hPhone, LPPHONESTATUS lpPhoneStatus)                                                                                                                                                                |
| uiuitui       | LONG | phoneGetVolume(HPHONE hPhone, DWORD dwHookSwitchDev, LPDWORD lpdwVolume)                                                                                                                                                   |
| tttatu        | LONG | phoneInitialize(LPHPHONEAPP lphPhoneApp, HINSTANCE hInstance, PHONECALLBACK lpfnCallback, LPCSTR lpszAppName, LPDWORD lpdwNumDevs)                                                                                         |
| tttattui      | LONG | phoneInitializeEx(LPHPHONEAPP lphPhoneApp, HINSTANCE hInstance, PHONECALLBACK lpfnCallback, LPCSTR lpszFriendlyAppName, LPDWORD lpdwNumDevs, LPDWORD lpdwAPIVersion, LPPHONEINITIALIZEEXPARAMS lpPhoneInitializeExParams)  |
| tttattui      | LONG | phoneInitializeExA(LPHPHONEAPP lphPhoneApp, HINSTANCE hInstance, PHONECALLBACK lpfnCallback, LPCSTR lpszFriendlyAppName, LPDWORD lpdwNumDevs, LPDWORD lpdwAPIVersion, LPPHONEINITIALIZEEXPARAMS lpPhoneInitializeExParams) |
| tttattui      | LONG | phoneInitializeExW(LPHPHONEAPP lphPhoneApp, HINSTANCE hInstance, PHONECALLBACK lpfnCallback, LPCSTR lpszFriendlyAppName, LPDWORD lpdwNumDevs, LPDWORD lpdwAPIVersion, LPPHONEINITIALIZEEXPARAMS lpPhoneInitializeExParams) |
| uiuiuiuitui   | LONG | phoneNegotiateAPIVersion(HPHONEAPP hPhoneApp, DWORD dwDeviceID, DWORD dwAPILowVersion, DWORD dwAPIHighVersion, LPDWORD lpdwAPIVersion, LPPHONEEXTENSIONID lpExtensionID)                                                   |
| uiuiuiuiuitui | LONG | phoneNegotiateExtVersion(HPHONEAPP hPhoneApp, DWORD dwDeviceID, DWORD dwAPIVersion, DWORD dwExtLowVersion, DWORD dwExtHighVersion, LPDWORD lpdwExtVersion)                                                                 |
| uiuituiuitui  | LONG | phoneOpen(HPHONEAPP hPhoneApp, DWORD dwDeviceID, LPHPHONE lphPhone, DWORD dwAPIVersion, DWORD dwExtVersion, DWORD_PTR dwCallbackInstance, DWORD dwPrivilege)                                                               |
| uiuitui       | LONG | phoneSetButtonInfo(HPHONE hPhone, DWORD dwButtonLampID, LPPHONEBUTTONINFO const lpButtonInfo)                                                                                                                              |
| uiuitui       | LONG | phoneSetButtonInfoA(HPHONE hPhone, DWORD dwButtonLampID, LPPHONEBUTTONINFO const lpButtonInfo)                                                                                                                             |

|             |      |                                                                                                                               |
|-------------|------|-------------------------------------------------------------------------------------------------------------------------------|
| uiuitui     | LONG | <a href="#">phoneSetButtonInfoW</a> (HPHONE hPhone, DWORD dwButtonLampID, LPPHONEBUTTONINFO const lpButtonInfo)               |
| uiuituiui   | LONG | <a href="#">phoneSetData</a> (HPHONE hPhone, DWORD dwDataID, LPVOID const lpData, DWORD dwSize)                               |
| uiuiuiauiui | LONG | <a href="#">phoneSetDisplay</a> (HPHONE hPhone, DWORD dwRow, DWORD dwColumn, LPCSTR lpsDisplay, DWORD dwSize)                 |
| uiuiuuiui   | LONG | <a href="#">phoneSetGain</a> (HPHONE hPhone, DWORD dwHookSwitchDev, DWORD dwGain)                                             |
| uiuiuuiui   | LONG | <a href="#">phoneSetHookSwitch</a> (HPHONE hPhone, DWORD dwHookSwitchDevs, DWORD dwHookSwitchMode)                            |
| uiuiuuiui   | LONG | <a href="#">phoneSetLamp</a> (HPHONE hPhone, DWORD dwButtonLampID, DWORD dwLampMode)                                          |
| uiuiuuiui   | LONG | <a href="#">phoneSetRing</a> (HPHONE hPhone, DWORD dwRingMode, DWORD dwVolume)                                                |
| uiuiuuiuiui | LONG | <a href="#">phoneSetStatusMessage</a> (HPHONE hPhone, DWORD dwPhoneStates, DWORD dwButtonModes, DWORD dwButtonStates)         |
| uiuiuuiui   | LONG | <a href="#">phoneSetVolume</a> (HPHONE hPhone, DWORD dwHookSwitchDev, DWORD dwVolume)                                         |
| uiui        | LONG | <a href="#">phoneShutdown</a> (HPHONEAPP hPhoneApp)                                                                           |
| aaui        | LONG | <a href="#">tapiGetLocationInfo</a> (LPSTR lpszCountryCode, LPSTR lpszCityCode)                                               |
| aaui        | LONG | <a href="#">tapiGetLocationInfoA</a> (LPSTR lpszCountryCode, LPSTR lpszCityCode)                                              |
| aaui        | LONG | <a href="#">tapiGetLocationInfoW</a> (LPSTR lpszCountryCode, LPSTR lpszCityCode)                                              |
| ttui        | LONG | <a href="#">tapiRequestDrop</a> (HWND hWnd, WPARAM wRequestID)                                                                |
| aaaaui      | LONG | <a href="#">tapiRequestMakeCall</a> (LPCSTR lpszDestAddress, LPCSTR lpszAppName, LPCSTR lpszCalledParty, LPCSTR lpszComment)  |
| aaaaui      | LONG | <a href="#">tapiRequestMakeCallA</a> (LPCSTR lpszDestAddress, LPCSTR lpszAppName, LPCSTR lpszCalledParty, LPCSTR lpszComment) |
| aaaaui      | LONG | <a href="#">tapiRequestMakeCallW</a> (LPCSTR lpszDestAddress, LPCSTR lpszAppName, LPCSTR lpszCalledParty, LPCSTR lpszComment) |

## User32.dll

|            |      |                                                                                                                                                                            |
|------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuit       | HKL  | ActivateKeyboardLayout(_In_ HKL hkl, _In_ UINT Flags)                                                                                                                      |
| ti         | BOOL | AddClipboardFormatListener(_In_ HWND hwnd)                                                                                                                                 |
| tuiii      | BOOL | AdjustWindowRect(_Inout_ LPRECT lpRect, _In_ DWORD dwStyle, _In_ BOOL bMenu)                                                                                               |
| tuiiii     | BOOL | AdjustWindowRectEx(_Inout_ LPRECT lpRect, _In_ DWORD dwStyle, _In_ BOOL bMenu, _In_ DWORD dwExStyle)                                                                       |
| uii        | BOOL | AllowSetForegroundWindow(_In_ DWORD dwProcessId)                                                                                                                           |
| tuiiii     | BOOL | AnimateWindow(_In_ HWND hwnd, _In_ DWORD dwTime, _In_ DWORD dwFlags)                                                                                                       |
| i          | BOOL | AnyPopup(void)                                                                                                                                                             |
| tuitsi     | BOOL | AppendMenu(_In_ HMENU hMenu, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCTSTR lpNewItem)                                                                       |
| tuitai     | BOOL | AppendMenuA(_In_ HMENU hMenu, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCSTR lpNewItem)                                                                       |
| tuitwi     | BOOL | AppendMenuW(_In_ HMENU hMenu, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCWSTR lpNewItem)                                                                      |
| tui        | UINT | ArrangeIconicWindows(_In_ HWND hwnd)                                                                                                                                       |
| uiuiii     | BOOL | AttachThreadInput(_In_ DWORD idAttach, _In_ DWORD idAttachTo, _In_ BOOL fAttach)                                                                                           |
| it         | HDWP | BeginDeferWindowPos(_In_ int nNumWindows)                                                                                                                                  |
| ttt        | HDC  | BeginPaint(_In_ HWND hwnd, _Out_ LPPAINTSTRUCT lpPaint)                                                                                                                    |
| ii         | BOOL | BlockInput(_In_ BOOL fBlockIt)                                                                                                                                             |
| ti         | BOOL | BringWindowToTop(_In_ HWND hwnd)                                                                                                                                           |
| uituittui  | LONG | BroadcastSystemMessage(_In_ DWORD dwFlags, _Inout_opt_ LPDWORD lpdwRecipients, _In_ UINT uiMessage, _In_ WPARAM wParam, _In_ LPARAM lParam)                                |
| uituittui  | LONG | BroadcastSystemMessageA(_In_ DWORD dwFlags, _Inout_opt_ LPDWORD lpdwRecipients, _In_ UINT uiMessage, _In_ WPARAM wParam, _In_ LPARAM lParam)                               |
| uituitttui | LONG | BroadcastSystemMessageEx(_In_ DWORD dwFlags, _Inout_opt_ LPDWORD lpdwRecipients, _In_ UINT uiMessage, _In_ WPARAM wParam, _In_ LPARAM lParam, _Out_opt_ PBSMINFO pBSMInfo) |
|            |      | BroadcastSystemMessageExA(_In_ DWORD dwFlags,                                                                                                                              |

|           |         |                                                                                                                                                                                          |
|-----------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uuiitttui | LONG    | <code>_Inout_opt_ LPDWORD lpdwRecipients, _In_ UINT uiMessage, _In_ WPARAM wParam, _In_ LPARAM lParam, _Out_opt_ PBSMINFO pBSMInfo)</code>                                               |
| uuiitttui | LONG    | <code>BroadcastSystemMessageExW(_In_ DWORD dwFlags, _Inout_opt_ LPDWORD lpdwRecipients, _In_ UINT uiMessage, _In_ WPARAM wParam, _In_ LPARAM lParam, _Out_opt_ PBSMINFO pBSMInfo)</code> |
| uuiitttui | LONG    | <code>BroadcastSystemMessageW(_In_ DWORD dwFlags, _Inout_opt_ LPDWORD lpdwRecipients, _In_ UINT uiMessage, _In_ WPARAM wParam, _In_ LPARAM lParam)</code>                                |
| ttuitti   | BOOL    | <code>CalculatePopupWindowPosition(_In_ const POINT *anchorPoint, _In_ const SIZE *windowSize, _In_ UINT flags, _In_opt_ RECT *excludeRect, _Out_ RECT *popupWindowPosition)</code>      |
| tii       | BOOL    | <code>CallMsgFilter(_In_ LPMSG lpMsg, _In_ int nCode)</code>                                                                                                                             |
| tii       | BOOL    | <code>CallMsgFilterA(_In_ LPMSG lpMsg, _In_ int nCode)</code>                                                                                                                            |
| tii       | BOOL    | <code>CallMsgFilterW(_In_ LPMSG lpMsg, _In_ int nCode)</code>                                                                                                                            |
| tittt     | LRESULT | <code>CallNextHookEx(_In_opt_ HHOOK hhk, _In_ int nCode, _In_ WPARAM wParam, _In_ LPARAM lParam)</code>                                                                                  |
| ttuittt   | LRESULT | <code>CallWindowProc(_In_ WNDPROC lpPrevWndFunc, _In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)</code>                                                           |
| ttuittt   | LRESULT | <code>CallWindowProcA(_In_ WNDPROC lpPrevWndFunc, _In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)</code>                                                          |
| ttuittt   | LRESULT | <code>CallWindowProcW(_In_ WNDPROC lpPrevWndFunc, _In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)</code>                                                          |
| tuituituh | WORD    | <code>CascadeWindows(_In_opt_ HWND hwndParent, _In_ UINT wHow, _In_opt_ const RECT *lpRect, _In_ UINT cKids, _In_opt_ const HWND *lpKids)</code>                                         |
| tii       | BOOL    | <code>ChangeClipboardChain(_In_ HWND hWndRemove, _In_ HWND hWndNewNext)</code>                                                                                                           |
| tuiui     | LONG    | <code>ChangeDisplaySettings(_In_ DEVMODE *lpDevMode, _In_ DWORD dwflags)</code>                                                                                                          |
| tuiui     | LONG    | <code>ChangeDisplaySettingsA(_In_ DEVMODE *lpDevMode, _In_ DWORD dwflags)</code>                                                                                                         |
| sttuitui  | LONG    | <code>ChangeDisplaySettingsEx(_In_ LPCTSTR lpszDeviceName, _In_ DEVMODE *lpDevMode, HWND hwnd, _In_ DWORD dwflags, _In_ LPVOID lParam)</code>                                            |
| attuitui  | LONG    | <code>ChangeDisplaySettingsExA(_In_ LPCSTR lpszDeviceName, _In_ DEVMODE *lpDevMode, HWND hwnd, _In_ DWORD dwflags, _In_ LPVOID lParam)</code>                                            |
|           |         |                                                                                                                                                                                          |

|          |        |                                                                                                                                        |
|----------|--------|----------------------------------------------------------------------------------------------------------------------------------------|
| wttuitui | LONG   | ChangeDisplaySettingsExW(_In_ LPCWSTR lpszDeviceName, _In_ DEVMODE *lpDevMode, HWND hwnd, _In_ DWORD dwflags, _In_ LPVOID lParam)      |
| tuiui    | LONG   | ChangeDisplaySettingsW(_In_ DEVMODE *lpDevMode, _In_ DWORD dwflags)                                                                    |
| uiiii    | BOOL   | ChangeWindowMessageFilter(_In_ UINT message, _In_ DWORD dwFlag)                                                                        |
| tuiuiti  | BOOL   | ChangeWindowMessageFilterEx(_In_ HWND hWnd, _In_ UINT message, _In_ DWORD action, _Inout_opt_ PCHANGEFILTERSTRUCT pChangeFilterStruct) |
| ss       | LPTSTR | CharLower(_Inout_ LPTSTR lpsz)                                                                                                         |
| aa       | LPSTR  | CharLowerA(_Inout_ LPSTR lpsz)                                                                                                         |
| suiii    | DWORD  | CharLowerBuf(_Inout_ LPTSTR lpsz, _In_ DWORD cchLength)                                                                                |
| aiiui    | DWORD  | CharLowerBuffA(_Inout_ LPSTR lpsz, _In_ DWORD cchLength)                                                                               |
| wuiii    | DWORD  | CharLowerBuffW(_Inout_ LPWSTR lpsz, _In_ DWORD cchLength)                                                                              |
| ww       | LPWSTR | CharLowerW(_Inout_ LPWSTR lpsz)                                                                                                        |
| ss       | LPTSTR | CharNext(_In_ LPCTSTR lpsz)                                                                                                            |
| aa       | LPSTR  | CharNextA(_In_ LPCSTR lpsz)                                                                                                            |
| uhauia   | LPSTR  | CharNextExA(_In_ WORD CodePage, _In_ LPCSTR lpCurrentChar, _In_ DWORD dwFlags)                                                         |
| ww       | LPWSTR | CharNextW(_In_ LPCWSTR lpsz)                                                                                                           |
| sss      | LPTSTR | CharPrev(_In_ LPCTSTR lpszStart, _In_ LPCTSTR lpszCurrent)                                                                             |
| aaa      | LPSTR  | CharPrevA(_In_ LPCSTR lpszStart, _In_ LPCSTR lpszCurrent)                                                                              |
| uhaauia  | LPSTR  | CharPrevExA(_In_ WORD CodePage, _In_ LPCSTR lpStart, _In_ LPCSTR lpCurrentChar, _In_ DWORD dwFlags)                                    |
| www      | LPWSTR | CharPrevW(_In_ LPCWSTR lpszStart, _In_ LPCWSTR lpszCurrent)                                                                            |
| sai      | BOOL   | CharToOem(_In_ LPCTSTR lpszSrc, _Out_ LPSTR lpszDst)                                                                                   |
| aa       | BOOL   | CharToOemA(_In_ LPCSTR lpszSrc, _Out_ LPSTR lpszDst)                                                                                   |
| sauii    | BOOL   | CharToOemBuff(_In_ LPCTSTR lpszSrc, _Out_ LPSTR lpszDst, _In_ DWORD cchDstLength)                                                      |
| aauii    | BOOL   | CharToOemBuffA(_In_ LPCSTR lpszSrc, _Out_ LPSTR lpszDst, _In_ DWORD cchDstLength)                                                      |
| wauii    | BOOL   | CharToOemBuffW(_In_ LPCWSTR lpszSrc, _Out_ LPSTR lpszDst, _In_ DWORD cchDstLength)                                                     |
|          |        |                                                                                                                                        |

|          |        |                                                                                                                |
|----------|--------|----------------------------------------------------------------------------------------------------------------|
| wai      | BOOL   | CharToOemW(_In_ LPCWSTR lpszSrc, _Out_ LPSTR lpszDst)                                                          |
| ss       | LPTSTR | CharUpper(_Inout_ LPTSTR lpsz)                                                                                 |
| aa       | LPSTR  | CharUpperA(_Inout_ LPSTR lpsz)                                                                                 |
| suiui    | DWORD  | CharUpperBuff(_Inout_ LPTSTR lpsz, _In_ DWORD cchLength)                                                       |
| aiuii    | DWORD  | CharUpperBuffA(_Inout_ LPSTR lpsz, _In_ DWORD cchLength)                                                       |
| wuiui    | DWORD  | CharUpperBuffW(_Inout_ LPWSTR lpsz, _In_ DWORD cchLength)                                                      |
| ww       | LPWSTR | CharUpperW(_Inout_ LPWSTR lpsz)                                                                                |
| tiuii    | BOOL   | CheckDlgButton(_In_ HWND hDlg, _In_ int nIDButton, _In_ UINT uCheck)                                           |
| tuiuii   | DWORD  | CheckMenuItem(_In_ HMENU hmenu, _In_ UINT uIDCheckItem, _In_ UINT uCheck)                                      |
| tuiuiuii | BOOL   | CheckMenuRadioItem(_In_ HMENU hmenu, _In_ UINT idFirst, _In_ UINT idLast, _In_ UINT idCheck, _In_ UINT uFlags) |
| tiii     | BOOL   | CheckRadioButton(_In_ HWND hDlg, _In_ int nIDFirstButton, _In_ int nIDLastButton, _In_ int nIDCheckButton)     |
| ti6t     | HWND   | ChildWindowFromPoint(_In_ HWND hWndParent, _In_ POINT Point)                                                   |
| ti6uit   | HWND   | ChildWindowFromPointEx(_In_ HWND hWndParent, _In_ POINT pt, _In_ UINT uFlags)                                  |
| tii      | BOOL   | ClientToScreen(_In_ HWND hWnd, _Inout_ LPPOINT lpPoint)                                                        |
| ti       | BOOL   | ClipCursor(_In_opt_ const RECT *lpRect)                                                                        |
| i        | BOOL   | CloseClipboard(void)                                                                                           |
| ti       | BOOL   | CloseDesktop(_In_ HDESK hDesktop)                                                                              |
| ti       | BOOL   | CloseGestureInfoHandle(HGESTUREINFO hGestureInfo)                                                              |
| ti       | BOOL   | CloseTouchInputHandle(_In_ HTOUCHINPUT hTouchInput)                                                            |
| ti       | BOOL   | CloseWindow(_In_ HWND hwnd)                                                                                    |
| ti       | BOOL   | CloseWindowStation(_In_ HWINSTA hWinSta)                                                                       |
| tii      | int    | CopyAcceleratorTable(_In_ HACCEL hAccelSrc, _Out_opt_ LPACCEL lpAccelDst, _In_ int cAccelEntries)              |
| tiii     | int    | CopyAcceleratorTableA(_In_ HACCEL hAccelSrc, _Out_opt_ LPACCEL lpAccelDst, _In_ int cAccelEntries)             |
| tiii     | int    | CopyAcceleratorTableW(_In_ HACCEL hAccelSrc, _Out_opt_ LPACCEL lpAccelDst, _In_ int cAccelEntries)             |

|             |         |                                                                                                                                                                                                                                                   |
|-------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tt          | HICON   | CopyIcon(_In_ HICON hIcon)                                                                                                                                                                                                                        |
| tuiiuit     | HANDLE  | CopyImage(_In_ HANDLE hImage, _In_ UINT uType, _In_ int cxDesired, _In_ int cyDesired, _In_ UINT fuFlags)                                                                                                                                         |
| tti         | BOOL    | CopyRect(_Out_ LPRECT lprcDst, _In_ const RECT *lprcSrc)                                                                                                                                                                                          |
| i           | int     | CountClipboardFormats(void)                                                                                                                                                                                                                       |
| tit         | HACCEL  | CreateAcceleratorTable(_In_ LPACCEL lpaccl, _In_ int cEntries)                                                                                                                                                                                    |
| tit         | HACCEL  | CreateAcceleratorTableA(_In_ LPACCEL lpaccl, _In_ int cEntries)                                                                                                                                                                                   |
| tit         | HACCEL  | CreateAcceleratorTableW(_In_ LPACCEL lpaccl, _In_ int cEntries)                                                                                                                                                                                   |
| tiii        | BOOL    | CreateCaret(_In_ HWND hWnd, _In_opt_ HBITMAP hBitmap, _In_ int nWidth, _In_ int nHeight)                                                                                                                                                          |
| tiiiiitt    | HCURSOR | CreateCursor(_In_opt_ HINSTANCE hInst, _In_ int xHotSpot, _In_ int yHotSpot, _In_ int nWidth, _In_ int nHeight, _In_ const VOID *pvANDPlane, _In_ const VOID *pvXORPlane)                                                                         |
| sstuiuit    | HDESK   | CreateDesktop(_In_ LPCTSTR lpszDesktop, _Reserved_ LPCTSTR lpszDevice, _Reserved_ DEVMODE *pDevmode, _In_ DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                   |
| aatuiuit    | HDESK   | CreateDesktopA(_In_ LPCSTR lpszDesktop, _Reserved_ LPCSTR lpszDevice, _Reserved_ DEVMODE *pDevmode, _In_ DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                    |
| sstuiuituit | HDESK   | CreateDesktopEx(_In_ LPCTSTR lpszDesktop, _Reserved_ LPCTSTR lpszDevice, _Reserved_ DEVMODE *pDevmode, _In_ DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa, _In_ ULONG ulHeapSize, _Reserved_ PVOID pvoid)  |
| aatuiuituit | HDESK   | CreateDesktopExA(_In_ LPCSTR lpszDesktop, _Reserved_ LPCSTR lpszDevice, _Reserved_ DEVMODE *pDevmode, _In_ DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa, _In_ ULONG ulHeapSize, _Reserved_ PVOID pvoid)   |
| wwtuiuituit | HDESK   | CreateDesktopExW(_In_ LPCWSTR lpszDesktop, _Reserved_ LPCWSTR lpszDevice, _Reserved_ DEVMODE *pDevmode, _In_ DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa, _In_ ULONG ulHeapSize, _Reserved_ PVOID pvoid) |
| wwtuiuit    | HDESK   | CreateDesktopW(_In_ LPCWSTR lpszDesktop, _Reserved_ LPCWSTR lpszDevice, _Reserved_ DEVMODE *pDevmode, _In_ DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                  |

|            |       |                                                                                                                                                                                                                                 |
|------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |       | dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                                                                                                                                           |
| ttttt      | HWND  | CreateDialogIndirectParam(_In_opt_ HINSTANCE hInstance, _In_ LPCDLGTEMPLATE lpTemplate, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM lParamInit)                                                        |
| ttttt      | HWND  | CreateDialogIndirectParamA(_In_opt_ HINSTANCE hInstance, _In_ LPCDLGTEMPLATE lpTemplate, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM lParamInit)                                                       |
| ttttt      | HWND  | CreateDialogIndirectParamW(_In_opt_ HINSTANCE hInstance, _In_ LPCDLGTEMPLATE lpTemplate, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM lParamInit)                                                       |
| tssttt     | HWND  | CreateDialogParam(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpTemplateName, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)                                                                  |
| tatttt     | HWND  | CreateDialogParamA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpTemplateName, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)                                                                  |
| twtttt     | HWND  | CreateDialogParamW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpTemplateName, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)                                                                 |
| tiucuctt   | HICON | CreateIcon(_In_opt_ HINSTANCE hInstance, _In_ int nWidth, _In_ int nHeight, _In_ BYTE cPlanes, _In_ BYTE cBitsPixel, _In_ const BYTE *lpbANDbits, _In_ const BYTE *lpbXORbits)                                                  |
| tuiiuit    | HICON | CreateIconFromResource(_In_ PBYTE presbits, _In_ DWORD dwResSize, _In_ BOOL fIcon, _In_ DWORD dwVer)                                                                                                                            |
| tuiuiiuit  | HICON | CreateIconFromResourceEx(_In_ PBYTE pbIconBits, _In_ DWORD cbIconBits, _In_ BOOL fIcon, _In_ DWORD dwVersion, _In_ int cxDesired, _In_ int cyDesired, _In_ UINT uFlags)                                                         |
| tt         | HICON | CreateIconIndirect(_In_ PICONINFO piconinfo)                                                                                                                                                                                    |
| ssuiiiittt | HWND  | CreateMDIWindow(_In_ LPCTSTR lpClassName, _In_ LPCTSTR lpWindowName, _In_ DWORD dwStyle, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight, _In_opt_ HWND hWndParent, _In_opt_ HINSTANCE hInstance, _In_ LPARAM lParam) |
| aauiiiittt | HWND  | CreateMDIWindowA(_In_ LPCSTR lpClassName, _In_ LPCSTR lpWindowName, _In_ DWORD dwStyle, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight, _In_opt_ HWND                                                                |

|               |          |                                                                                                                                                                                                                                                                                          |
|---------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |          | hWndParent, _In_opt_ HINSTANCE hInstance, _In_ LPARAM lParam)                                                                                                                                                                                                                            |
| wwuiiiitttt   | HWND     | CreateMDIWindowW(_In_ LPCWSTR lpClassName, _In_ LPCWSTR lpWindowName, _In_ DWORD dwStyle, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight, _In_opt_ HWND hWndParent, _In_opt_ HINSTANCE hInstance, _In_ LPARAM lParam)                                                         |
| t             | HMENU    | CreateMenu(void)                                                                                                                                                                                                                                                                         |
| t             | HMENU    | CreatePopupMenu(void)                                                                                                                                                                                                                                                                    |
| uissuiiiitttt | HWND     | CreateWindowEx(_In_ DWORD dwExStyle, _In_opt_ LPCTSTR lpClassName, _In_opt_ LPCTSTR lpWindowName, _In_ DWORD dwStyle, _In_ int x, _In_ int y, _In_ int nWidth, _In_ int nHeight, _In_opt_ HWND hWndParent, _In_opt_ HMENU hMenu, _In_opt_ HINSTANCE hInstance, _In_opt_ LPVOID lpParam)  |
| uiauiiiitttt  | HWND     | CreateWindowExA(_In_ DWORD dwExStyle, _In_opt_ LPCSTR lpClassName, _In_opt_ LPCSTR lpWindowName, _In_ DWORD dwStyle, _In_ int x, _In_ int y, _In_ int nWidth, _In_ int nHeight, _In_opt_ HWND hWndParent, _In_opt_ HMENU hMenu, _In_opt_ HINSTANCE hInstance, _In_opt_ LPVOID lpParam)   |
| uiwwuiiiitttt | HWND     | CreateWindowExW(_In_ DWORD dwExStyle, _In_opt_ LPCWSTR lpClassName, _In_opt_ LPCWSTR lpWindowName, _In_ DWORD dwStyle, _In_ int x, _In_ int y, _In_ int nWidth, _In_ int nHeight, _In_opt_ HWND hWndParent, _In_opt_ HMENU hMenu, _In_opt_ HINSTANCE hInstance, _In_opt_ LPVOID lpParam) |
| suiiutt       | HWINSTA  | CreateWindowStation(_In_opt_ LPCTSTR lpwinsta, DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                                                                                                                     |
| aiuiutt       | HWINSTA  | CreateWindowStationA(_In_opt_ LPCSTR lpwinsta, DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                                                                                                                     |
| wuiiutt       | HWINSTA  | CreateWindowStationW(_In_opt_ LPCWSTR lpwinsta, DWORD dwFlags, _In_ ACCESS_MASK dwDesiredAccess, _In_opt_ LPSECURITY_ATTRIBUTES lpsa)                                                                                                                                                    |
| uituii        | BOOL     | DdeAbandonTransaction(_In_ DWORD idInst, _In_ HCONV hConv, _In_ DWORD idTransaction)                                                                                                                                                                                                     |
| ttt           | LPBYTE   | DdeAccessData(_In_ HDDEDATA hData, _Out_opt_ LPDWORD pcbDataSize)                                                                                                                                                                                                                        |
| ttuiuit       | HDDEDATA | DdeAddData(_In_ HDDEDATA hData, _In_ LPBYTE pSrc, _In_ DWORD cb, _In_ DWORD cbOff)                                                                                                                                                                                                       |
| tuittuiiuit   | HDDEDATA | DdeClientTransaction(_In_opt_ LPBYTE pData, _In_ DWORD cbData, _In_ HCONV hConv, _In_opt_ HSZ hszItem, _In_ UINT wFmt, _In_ UINT wType, _In_ DWORD                                                                                                                                       |

|               |           |                                                                                                                                                      |
|---------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |           | dwTimeout, _Out_opt_ LPDWORD pdwResult)                                                                                                              |
| tii           | int       | DdeCompStringHandles(_In_ HSZ hsz1, _In_ HSZ hsz2)                                                                                                   |
| uiittt        | HCONV     | DdeConnect(_In_ DWORD idInst, _In_ HSZ hszService, _In_ HSZ hszTopic, _In_opt_ PCONVCONTEXT pCC)                                                     |
| uiitttt       | HCONVLIST | DdeConnectList(_In_ DWORD idInst, _In_ HSZ hszService, _In_ HSZ hszTopic, _In_ HCONVLIST hConvList, _In_opt_ PCONVCONTEXT pCC)                       |
| uituiuituiuit | HDDEDATA  | DdeCreateDataHandle(_In_ DWORD idInst, _In_opt_ LPBYTE pSrc, _In_ DWORD cb, _In_ DWORD cbOff, _In_opt_ HSZ hszItem, _In_ UINT wFmt, _In_ UINT afCmd) |
| uisit         | HSZ       | DdeCreateStringHandle(_In_ DWORD idInst, _In_ LPTSTR psz, _In_ int iCodePage)                                                                        |
| uiait         | HSZ       | DdeCreateStringHandleA(_In_ DWORD idInst, _In_ LPSTR psz, _In_ int iCodePage)                                                                        |
| uiwit         | HSZ       | DdeCreateStringHandleW(_In_ DWORD idInst, _In_ LPWSTR psz, _In_ int iCodePage)                                                                       |
| ti            | BOOL      | DdeDisconnect(_In_ HCONV hConv)                                                                                                                      |
| ti            | BOOL      | DdeDisconnectList(_In_ HCONVLIST hConvList)                                                                                                          |
| uituii        | BOOL      | DdeEnableCallback(_In_ DWORD idInst, _In_ HCONV hConv, _In_ UINT wCmd)                                                                               |
| ti            | BOOL      | DdeFreeDataHandle(_In_ HDDEDATA hData)                                                                                                               |
| uiti          | BOOL      | DdeFreeStringHandle(_In_ DWORD idInst, _In_ HSZ hsz)                                                                                                 |
| ttuiuiui      | DWORD     | DdeGetData(_In_ HDDEDATA hData, _Out_opt_ LPBYTE pDst, _In_ DWORD cbMax, _In_ DWORD cbOff)                                                           |
| uiui          | UINT      | DdeGetLastError(_In_ DWORD idInst)                                                                                                                   |
| ti            | BOOL      | DdeImpersonateClient(_In_ HCONV hConv)                                                                                                               |
| ttuiuiui      | UINT      | DdeInitialize(_Inout_ LPDWORD pidInst, _In_ PFNCALLBACK pfnCallback, _In_ DWORD afCmd, _Reserved_ DWORD ulRes)                                       |
| ttuiuiui      | UINT      | DdeInitializeA(_Inout_ LPDWORD pidInst, _In_ PFNCALLBACK pfnCallback, _In_ DWORD afCmd, _Reserved_ DWORD ulRes)                                      |
| ttuiuiui      | UINT      | DdeInitializeW(_Inout_ LPDWORD pidInst, _In_ PFNCALLBACK pfnCallback, _In_ DWORD afCmd, _Reserved_ DWORD ulRes)                                      |
| uiti          | BOOL      | DdeKeepStringHandle(_In_ DWORD idInst, _In_ HSZ hsz)                                                                                                 |
| uittuit       | HDDEDATA  | DdeNameService(_In_ DWORD idInst, _In_opt_ HSZ hsz1, _In_opt_ HSZ hsz2, _In_ UINT afCmd)                                                             |
| uitti         | BOOL      | DdePostAdvise(_In_ DWORD idInst, _In_ HSZ hszTopic, _In_ HSZ hszItem)                                                                                |
|               |           | DdeQueryConvInfo(_In_ HCONV hConv, _In_ DWORD                                                                                                        |

|           |         |                                                                                                                                                          |
|-----------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitui    | UINT    | idTransaction, _Inout_ PCONVINFO pConvInfo)                                                                                                              |
| ttt       | HCONV   | DdeQueryNextServer(_In_ HCONVLIST hConvList, _In_ HCONV hConvPrev)                                                                                       |
| uitsuiiui | DWORD   | DdeQueryString(_In_ DWORD idInst, _In_ HSZ hsz, _Out_opt_ LPTSTR psz, _In_ DWORD cchMax, _In_ int iCodePage)                                             |
| uitaiiui  | DWORD   | DdeQueryStringA(_In_ DWORD idInst, _In_ HSZ hsz, _Out_opt_ LPSTR psz, _In_ DWORD cchMax, _In_ int iCodePage)                                             |
| uitwuiiui | DWORD   | DdeQueryStringW(_In_ DWORD idInst, _In_ HSZ hsz, _Out_opt_ LPWSTR psz, _In_ DWORD cchMax, _In_ int iCodePage)                                            |
| tt        | HCONV   | DdeReconnect(_In_ HCONV hConv)                                                                                                                           |
| ttti      | BOOL    | DdeSetQualityOfService(_In_ HWND hwndClient, _In_ const SECURITY_QUALITY_OF_SERVICE *pqosNew, _Out_ PSECURITY_QUALITY_OF_SERVICE pqosPrev)               |
| tuiti     | BOOL    | DdeSetUserHandle(_In_ HCONV hConv, _In_ DWORD id, _In_ DWORD_PTR hUser)                                                                                  |
| ti        | BOOL    | DdeUnaccessData(_In_ HDDEDATA hData)                                                                                                                     |
| uii       | BOOL    | DdeUninitialize(_In_ DWORD idInst)                                                                                                                       |
| tuittt    | LRESULT | DefDlgProc(_In_ HWND hDlg, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                        |
| tuittt    | LRESULT | DefDlgProcA(_In_ HWND hDlg, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                       |
| tuittt    | LRESULT | DefDlgProcW(_In_ HWND hDlg, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                       |
| ttiiiiuit | HDWP    | DeferWindowPos(_In_ HDWP hWinPosInfo, _In_ HWND hWnd, _In_opt_ HWND hWndInsertAfter, _In_ int x, _In_ int y, _In_ int cx, _In_ int cy, _In_ UINT uFlags) |
| ttuittt   | LRESULT | DefFrameProc(_In_ HWND hWnd, _In_ HWND hWndMDIClient, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                            |
| ttuittt   | LRESULT | DefFrameProcA(_In_ HWND hWnd, _In_ HWND hWndMDIClient, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                           |
| ttuittt   | LRESULT | DefFrameProcW(_In_ HWND hWnd, _In_ HWND hWndMDIClient, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                           |
| tuittt    | LRESULT | DefMDIChildProc(_In_ HWND hWnd, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                  |
| tuittt    | LRESULT | DefMDIChildProcA(_In_ HWND hWnd, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                 |

|        |         |                                                                                                                                                                              |
|--------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuittt | LRESULT | DefMDIChildProcW(_In_ HWND hWnd, _In_ UINT uMsg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                     |
| tiuit  | LRESULT | DefRawInputProc(_In_ PRAWINPUT *paRawInput, _In_ INT nInput, _In_ UINT cbSizeHeader)                                                                                         |
| tuittt | LRESULT | DefWindowProc(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                         |
| tuittt | LRESULT | DefWindowProcA(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                        |
| tuittt | LRESULT | DefWindowProcW(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                        |
| tuiiii | BOOL    | DeleteMenu(_In_ HMENU hMenu, _In_ UINT uPosition, _In_ UINT uFlags)                                                                                                          |
| ti     | BOOL    | DeregisterShellHookWindow(_In_ HWND hwnd)                                                                                                                                    |
| ti     | BOOL    | DestroyAcceleratorTable(_In_ HACCEL hAccel)                                                                                                                                  |
| i      | BOOL    | DestroyCaret(void)                                                                                                                                                           |
| ti     | BOOL    | DestroyCursor(_In_ HCURSOR hCursor)                                                                                                                                          |
| ti     | BOOL    | DestroyIcon(_In_ HICON hIcon)                                                                                                                                                |
| ti     | BOOL    | DestroyMenu(_In_ HMENU hMenu)                                                                                                                                                |
| ti     | BOOL    | DestroyWindow(_In_ HWND hwnd)                                                                                                                                                |
| tttttt | INT_PTR | DialogBoxIndirectParam(_In_opt_ HINSTANCE hInstance, _In_ LPCDLGTEMPLATE hDialogTemplate, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)  |
| tttttt | INT_PTR | DialogBoxIndirectParamA(_In_opt_ HINSTANCE hInstance, _In_ LPCDLGTEMPLATE hDialogTemplate, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam) |
| tttttt | INT_PTR | DialogBoxIndirectParamW(_In_opt_ HINSTANCE hInstance, _In_ LPCDLGTEMPLATE hDialogTemplate, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam) |
| tsTTTT | INT_PTR | DialogBoxParam(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpTemplateName, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)                  |
| tatttt | INT_PTR | DialogBoxParamA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpTemplateName, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)                  |
| twTTTT | INT_PTR | DialogBoxParamW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpTemplateName, _In_opt_ HWND hWndParent, _In_opt_ DLGPROC lpDialogFunc, _In_ LPARAM dwInitParam)                 |

|          |         |                                                                                                                                   |
|----------|---------|-----------------------------------------------------------------------------------------------------------------------------------|
|          |         | dwInitParam)                                                                                                                      |
| tt       | LRESULT | DispatchMessage(_In_ const MSG *lpmsg)                                                                                            |
| tt       | LRESULT | DispatchMessageA(_In_ const MSG *lpmsg)                                                                                           |
| tt       | LRESULT | DispatchMessageW(_In_ const MSG *lpmsg)                                                                                           |
| tsiiiii  | int     | DlgDirList(_In_ HWND hDlg, _Inout_ LPTSTR lpPathSpec, _In_ int nIDListBox, _In_ int nIDStaticPath, _In_ UINT uFileType)           |
| taiiiiii | int     | DlgDirListA(_In_ HWND hDlg, _Inout_ LPSTR lpPathSpec, _In_ int nIDListBox, _In_ int nIDStaticPath, _In_ UINT uFileType)           |
| tsiiiii  | int     | DlgDirListComboBox(_In_ HWND hDlg, _Inout_ LPTSTR lpPathSpec, _In_ int nIDComboBox, _In_ int nIDStaticPath, _In_ UINT uFileType)  |
| taiiiiii | int     | DlgDirListComboBoxA(_In_ HWND hDlg, _Inout_ LPSTR lpPathSpec, _In_ int nIDComboBox, _In_ int nIDStaticPath, _In_ UINT uFileType)  |
| twiiiii  | int     | DlgDirListComboBoxW(_In_ HWND hDlg, _Inout_ LPWSTR lpPathSpec, _In_ int nIDComboBox, _In_ int nIDStaticPath, _In_ UINT uFileType) |
| twiiiii  | int     | DlgDirListW(_In_ HWND hDlg, _Inout_ LPWSTR lpPathSpec, _In_ int nIDListBox, _In_ int nIDStaticPath, _In_ UINT uFileType)          |
| tsiii    | BOOL    | DlgDirSelectComboBoxEx(_In_ HWND hDlg, _Out_ LPTSTR lpString, _In_ int nCount, _In_ int nIDComboBox)                              |
| taiiii   | BOOL    | DlgDirSelectComboBoxExA(_In_ HWND hDlg, _Out_ LPSTR lpString, _In_ int nCount, _In_ int nIDComboBox)                              |
| twiii    | BOOL    | DlgDirSelectComboBoxExW(_In_ HWND hDlg, _Out_ LPWSTR lpString, _In_ int nCount, _In_ int nIDComboBox)                             |
| tsiii    | BOOL    | DlgDirSelectEx(_In_ HWND hDlg, _Out_ LPTSTR lpString, _In_ int nCount, _In_ int nIDListBox)                                       |
| taiiii   | BOOL    | DlgDirSelectExA(_In_ HWND hDlg, _Out_ LPSTR lpString, _In_ int nCount, _In_ int nIDListBox)                                       |
| twiii    | BOOL    | DlgDirSelectExW(_In_ HWND hDlg, _Out_ LPWSTR lpString, _In_ int nCount, _In_ int nIDListBox)                                      |
| ti6i     | BOOL    | DragDrect(_In_ HWND hwnd, _In_ POINT pt)                                                                                          |
| titti    | BOOL    | DrawAnimatedRects(_In_ HWND hwnd, _In_ int idAni, const RECT *lprcFrom, const RECT *lprcTo)                                       |
| ttuiii   | BOOL    | DrawCaption(_In_ HWND hwnd, _In_ HDC hdc, _In_ LPCRECT lprc, _In_ UINT uFlags)                                                    |
| ttuiiii  | BOOL    | DrawEdge(_In_ HDC hdc, _Inout_ LPRECT qrc, _In_ UINT edge, _In_ UINT grfFlags)                                                    |

|             |      |                                                                                                                                                                                                            |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii         | BOOL | <code>DrawFocusRect(_In_ HDC hdc, _In_ const RECT *lprc)</code>                                                                                                                                            |
| ttuiiii     | BOOL | <code>DrawFrameControl(_In_ HDC hdc, _In_ LPRECT lprc, _In_ UINT uType, _In_ UINT uState)</code>                                                                                                           |
| tiiti       | BOOL | <code>DrawIcon(_In_ HDC hdc, _In_ int X, _In_ int Y, _In_ HICON hIcon)</code>                                                                                                                              |
| tiitiiituii | BOOL | <code>DrawIconEx(_In_ HDC hdc, _In_ int xLeft, _In_ int yTop, _In_ HICON hIcon, _In_ int cxWidth, _In_ int cyWidth, _In_ UINT istepIfAniCur, _In_opt_ HBRUSH hbrFlickerFreeDraw, _In_ UINT diFlags)</code> |
| ti          | BOOL | <code>DrawMenuBar(_In_ HWND hwnd)</code>                                                                                                                                                                   |
| ttttiiiiiii | BOOL | <code>DrawState(_In_ HDC hdc, _In_ HBRUSH hbr, _In_ DRAWSTATEPROC lpOutputFunc, _In_ LPARAM lData, _In_ WPARAM wParam, _In_ int x, _In_ int y, _In_ int cx, _In_ int cy, _In_ UINT fuFlags)</code>         |
| ttttiiiiiii | BOOL | <code>DrawStateA(_In_ HDC hdc, _In_ HBRUSH hbr, _In_ DRAWSTATEPROC lpOutputFunc, _In_ LPARAM lData, _In_ WPARAM wParam, _In_ int x, _In_ int y, _In_ int cx, _In_ int cy, _In_ UINT fuFlags)</code>        |
| ttttiiiiiii | BOOL | <code>DrawStateW(_In_ HDC hdc, _In_ HBRUSH hbr, _In_ DRAWSTATEPROC lpOutputFunc, _In_ LPARAM lData, _In_ WPARAM wParam, _In_ int x, _In_ int y, _In_ int cx, _In_ int cy, _In_ UINT fuFlags)</code>        |
| tsituii     | int  | <code>DrawText(_In_ HDC hdc, _Inout_ LPCTSTR lpchText, _In_ int nCount, _Inout_ LPRECT lpRect, _In_ UINT uFormat)</code>                                                                                   |
| taituii     | int  | <code>DrawTextA(_In_ HDC hdc, _Inout_ LPCSTR lpchText, _In_ int nCount, _Inout_ LPRECT lpRect, _In_ UINT uFormat)</code>                                                                                   |
| tsituiti    | int  | <code>DrawTextEx(_In_ HDC hdc, _Inout_ LPTSTR lpchText, _In_ int cchText, _Inout_ LPRECT lprc, _In_ UINT dwDTFormat, _In_ LPDRAWTEXTPARAMS lpDTParams)</code>                                              |
| taituiti    | int  | <code>DrawTextExA(_In_ HDC hdc, _Inout_ LPSTR lpchText, _In_ int cchText, _Inout_ LPRECT lprc, _In_ UINT dwDTFormat, _In_ LPDRAWTEXTPARAMS lpDTParams)</code>                                              |
| twituiti    | int  | <code>DrawTextExW(_In_ HDC hdc, _Inout_ LPWSTR lpchText, _In_ int cchText, _Inout_ LPRECT lprc, _In_ UINT dwDTFormat, _In_ LPDRAWTEXTPARAMS lpDTParams)</code>                                             |
| twituii     | int  | <code>DrawTextW(_In_ HDC hdc, _Inout_ LPCWSTR lpchText, _In_ int nCount, _Inout_ LPRECT lpRect, _In_ UINT uFormat)</code>                                                                                  |
| i           | BOOL | <code>EmptyClipboard(void)</code>                                                                                                                                                                          |
| tuiiii      | BOOL | <code>EnableMenuItem(_In_ HMENU hMenu, _In_ UINT uIDEnableItem, _In_ UINT uEnable)</code>                                                                                                                  |
| tuiiii      | BOOL | <code>EnableScrollBar(_In_ HWND hwnd, _In_ UINT wSBflags, _In_ UINT wArrows)</code>                                                                                                                        |

|         |      |                                                                                                                           |
|---------|------|---------------------------------------------------------------------------------------------------------------------------|
| tii     | BOOL | EnableWindow(_In_ HWND hWnd, _In_ BOOL bEnable)                                                                           |
| ti      | BOOL | EndDeferWindowPos(_In_ HDWP hWinPosInfo)                                                                                  |
| ttd     | BOOL | EndDialog(_In_ HWND hDlg, _In_ INT_PTR nResult)                                                                           |
| i       | BOOL | EndMenu(void)                                                                                                             |
| ttd     | BOOL | EndPoint(_In_ HWND hWnd, _In_ const PAINTSTRUCT *lpPaint)                                                                 |
| tiii    | BOOL | EndTask(_In_ HWND hWnd, _In_ BOOL fShutDown, _In_ BOOL fForce)                                                            |
| ttd     | BOOL | EnumChildWindows(_In_opt_ HWND hWndParent, _In_ WNDENUMPROC lpEnumFunc, _In_ LPARAM lParam)                               |
| uiui    | UINT | EnumClipboardFormats(_In_ UINT format)                                                                                    |
| ttd     | BOOL | EnumDesktops(_In_opt_ HWINSTA hwinsta, _In_ DESKTOPENUMPROC lpEnumFunc, _In_ LPARAM lParam)                               |
| ttd     | BOOL | EnumDesktopsA(_In_opt_ HWINSTA hwinsta, _In_ DESKTOPENUMPROC lpEnumFunc, _In_ LPARAM lParam)                              |
| ttd     | BOOL | EnumDesktopsW(_In_opt_ HWINSTA hwinsta, _In_ DESKTOPENUMPROC lpEnumFunc, _In_ LPARAM lParam)                              |
| ttd     | BOOL | EnumDesktopWindows(_In_opt_ HDESK hDesktop, _In_ WNDENUMPROC lpfnc, _In_ LPARAM lParam)                                   |
| suituii | BOOL | EnumDisplayDevices(_In_ LPCTSTR lpDevice, _In_ DWORD iDevNum, _Out_ PDISPLAY_DEVICE lpDisplayDevice, _In_ DWORD dwFlags)  |
| autuii  | BOOL | EnumDisplayDevicesA(_In_ LPCSTR lpDevice, _In_ DWORD iDevNum, _Out_ PDISPLAY_DEVICE lpDisplayDevice, _In_ DWORD dwFlags)  |
| wuituii | BOOL | EnumDisplayDevicesW(_In_ LPCWSTR lpDevice, _In_ DWORD iDevNum, _Out_ PDISPLAY_DEVICE lpDisplayDevice, _In_ DWORD dwFlags) |
| tttd    | BOOL | EnumDisplayMonitors(_In_ HDC hdc, _In_ LPCRECT lprcClip, _In_ MONITORENUMPROC lpfncEnum, _In_ LPARAM dwData)              |
| suiti   | BOOL | EnumDisplaySettings(_In_ LPCTSTR lpszDeviceName, _In_ DWORD iModeNum, _Out_ DEVMODE *lpDevMode)                           |
| auti    | BOOL | EnumDisplaySettingsA(_In_ LPCSTR lpszDeviceName, _In_ DWORD iModeNum, _Out_ DEVMODE *lpDevMode)                           |
| suituii | BOOL | EnumDisplaySettingsEx(_In_ LPCTSTR lpszDeviceName, _In_ DWORD iModeNum, _Out_ DEVMODE *lpDevMode, _In_ DWORD dwFlags)     |
| autuii  | BOOL | EnumDisplaySettingsExA(_In_ LPCSTR lpszDeviceName, _In_ DWORD iModeNum, _Out_ DEVMODE *lpDevMode, _In_ DWORD dwFlags)     |
|         |      | EnumDisplaySettingsExW(_In_ LPCWSTR lpszDeviceName,                                                                       |

|         |      |                                                                                                                               |
|---------|------|-------------------------------------------------------------------------------------------------------------------------------|
| wuituii | BOOL | _In_ DWORD iModeNum, _Out_ DEVMODE *lpDevMode, _In_ DWORD dwFlags)                                                            |
| wuiti   | BOOL | EnumDisplaySettingsW(_In_ LPCWSTR lpszDeviceName, _In_ DWORD iModeNum, _Out_ DEVMODE *lpDevMode)                              |
| titi    | int  | EnumProps(_In_ HWND hWnd, _In_ PROPENUMPROC lpEnumFunc)                                                                       |
| titi    | int  | EnumPropsA(_In_ HWND hWnd, _In_ PROPENUMPROC lpEnumFunc)                                                                      |
| titi    | int  | EnumPropsEx(_In_ HWND hWnd, _In_ PROPENUMPROCEX lpEnumFunc, _In_ LPARAM lParam)                                               |
| titi    | int  | EnumPropsExA(_In_ HWND hWnd, _In_ PROPENUMPROCEX lpEnumFunc, _In_ LPARAM lParam)                                              |
| titi    | int  | EnumPropsExW(_In_ HWND hWnd, _In_ PROPENUMPROCEX lpEnumFunc, _In_ LPARAM lParam)                                              |
| titi    | int  | EnumPropsW(_In_ HWND hWnd, _In_ PROPENUMPROC lpEnumFunc)                                                                      |
| uitti   | BOOL | EnumThreadWindows(_In_ DWORD dwThreadId, _In_ WNDENUMPROC lpfnc, _In_ LPARAM lParam)                                          |
| titi    | BOOL | EnumWindows(_In_ WNDENUMPROC lpEnumFunc, _In_ LPARAM lParam)                                                                  |
| titi    | BOOL | EnumWindowStations(_In_ WINSTAENUMPROC lpEnumFunc, _In_ LPARAM lParam)                                                        |
| titi    | BOOL | EnumWindowStationsA(_In_ WINSTAENUMPROC lpEnumFunc, _In_ LPARAM lParam)                                                       |
| titi    | BOOL | EnumWindowStationsW(_In_ WINSTAENUMPROC lpEnumFunc, _In_ LPARAM lParam)                                                       |
| titi    | BOOL | EqualRect(_In_ const RECT *lprc1, _In_ const RECT *lprc2)                                                                     |
| titi    | int  | ExcludeUpdateRgn(_In_ HDC hDC, _In_ HWND hWnd)                                                                                |
| uiuii   | BOOL | ExitWindowsEx(_In_ UINT uFlags, _In_ DWORD dwReason)                                                                          |
| titi    | int  | FillRect(_In_ HDC hDC, _In_ const RECT *lprc, _In_ HBRUSH hbr)                                                                |
| sst     | HWND | FindWindow(_In_opt_ LPCTSTR lpClassName, _In_opt_ LPCTSTR lpWindowName)                                                       |
| aat     | HWND | FindWindowA(_In_opt_ LPCSTR lpClassName, _In_opt_ LPCSTR lpWindowName)                                                        |
| ttsst   | HWND | FindWindowEx(_In_opt_ HWND hwndParent, _In_opt_ HWND hwndChildAfter, _In_opt_ LPCTSTR lpszClass, _In_opt_ LPCTSTR lpszWindow) |
| ttaat   | HWND | FindWindowExA(_In_opt_ HWND hwndParent, _In_opt_ HWND hwndChildAfter, _In_opt_ LPCSTR lpszClass, _In_opt_ LPCSTR lpszWindow)  |
|         |      |                                                                                                                               |

|         |       |                                                                                                                                   |
|---------|-------|-----------------------------------------------------------------------------------------------------------------------------------|
| ttwwt   | HWND  | FindWindowExW(_In_opt_ HWND hwndParent, _In_opt_ HWND hwndChildAfter, _In_opt_ LPCWSTR lpszClass, _In_opt_ LPCWSTR lpszWindow)    |
| wwt     | HWND  | FindWindowW(_In_opt_ LPCWSTR lpClassName, _In_opt_ LPCWSTR lpWindowName)                                                          |
| tii     | BOOL  | FlashWindow(_In_ HWND hwnd, _In_ BOOL bInvert)                                                                                    |
| ti      | BOOL  | FlashWindowEx(_In_ PFLASHWINFO pfw)                                                                                               |
| titi    | int   | FrameRect(_In_ HDC hDC, _In_ const RECT *lprc, _In_ HBRUSH hbr)                                                                   |
| uiti    | BOOL  | FreeDDEIParam(_In_ UINT msg, _In_ LPARAM lParam)                                                                                  |
| t       | HWND  | GetActiveWindow(void)                                                                                                             |
| titsuii | BOOL  | GetAltTabInfo(_In_opt_ HWND hwnd, _In_ int iItem, _Inout_ PALTTABINFO pati, _Out_opt_ LPTSTR pszItemText, _In_ UINT cchItemText)  |
| titauii | BOOL  | GetAltTabInfoA(_In_opt_ HWND hwnd, _In_ int iItem, _Inout_ PALTTABINFO pati, _Out_opt_ LPSTR pszItemText, _In_ UINT cchItemText)  |
| titwuii | BOOL  | GetAltTabInfoW(_In_opt_ HWND hwnd, _In_ int iItem, _Inout_ PALTTABINFO pati, _Out_opt_ LPWSTR pszItemText, _In_ UINT cchItemText) |
| tuit    | HWND  | GetAncestor(_In_ HWND hwnd, _In_ UINT gaFlags)                                                                                    |
| ih      | SHORT | GetAsyncKeyState(_In_ int vKey)                                                                                                   |
| t       | HWND  | GetCapture(void)                                                                                                                  |
| ui      | UINT  | GetCaretBlinkTime(void)                                                                                                           |
| ti      | BOOL  | GetCaretPos(_Out_ LPPOINT lpPoint)                                                                                                |
| tsti    | BOOL  | GetClassInfo(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpClassName, _Out_ LPWNDCLASS lpWndClass)                                 |
| tati    | BOOL  | GetClassInfoA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpClassName, _Out_ LPWNDCLASS lpWndClass)                                 |
| tsti    | BOOL  | GetClassInfoEx(_In_opt_ HINSTANCE hinst, _In_ LPCTSTR lpszClass, _Out_ LPWNDCLASSEX lpwecx)                                       |
| tati    | BOOL  | GetClassInfoExA(_In_opt_ HINSTANCE hinst, _In_ LPCSTR lpszClass, _Out_ LPWNDCLASSEX lpwecx)                                       |
| twti    | BOOL  | GetClassInfoExW(_In_opt_ HINSTANCE hinst, _In_ LPCWSTR lpszClass, _Out_ LPWNDCLASSEX lpwecx)                                      |
| twti    | BOOL  | GetClassInfoW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpClassName, _Out_ LPWNDCLASS lpWndClass)                                |
| tiui    | DWORD | GetClassLong(_In_ HWND hwnd, _In_ int nIndex)                                                                                     |
| tiui    | DWORD | GetClassLongA(_In_ HWND hwnd, _In_ int nIndex)                                                                                    |

|        |         |                                                                                                     |
|--------|---------|-----------------------------------------------------------------------------------------------------|
| tiui   | DWORD   | GetClassLongW(_In_ HWND hWnd, _In_ int nIndex)                                                      |
| tsii   | int     | GetClassName(_In_ HWND hWnd, _Out_ LPTSTR lpClassName, _In_ int nMaxCount)                          |
| taii   | int     | GetClassNameA(_In_ HWND hWnd, _Out_ LPSTR lpClassName, _In_ int nMaxCount)                          |
| twii   | int     | GetClassNameW(_In_ HWND hWnd, _Out_ LPWSTR lpClassName, _In_ int nMaxCount)                         |
| tiuh   | WORD    | GetClassWord(_In_ HWND hWnd, _In_ int nIndex)                                                       |
| tii    | BOOL    | GetClientRect(_In_ HWND hWnd, _Out_ LPRECT lpRect)                                                  |
| uit    | HANDLE  | GetClipboardData(_In_ UINT uFormat)                                                                 |
| uisii  | int     | GetClipboardFormatName(_In_ UINT format, _Out_ LPTSTR lpszFormatName, _In_ int cchMaxCount)         |
| uiaii  | int     | GetClipboardFormatNameA(_In_ UINT format, _Out_ LPSTR lpszFormatName, _In_ int cchMaxCount)         |
| uiwii  | int     | GetClipboardFormatNameW(_In_ UINT format, _Out_ LPWSTR lpszFormatName, _In_ int cchMaxCount)        |
| t      | HWND    | GetClipboardOwner(void)                                                                             |
| ui     | DWORD   | GetClipboardSequenceNumber(void)                                                                    |
| t      | HWND    | GetClipboardViewer(void)                                                                            |
| ti     | BOOL    | GetClipCursor(_Out_ LPRECT lpRect)                                                                  |
| tii    | BOOL    | GetComboBoxInfo(_In_ HWND hwndCombo, _Out_ PCOMBOBOXINFO pcbi)                                      |
| t      | HCURSOR | GetCursor(void)                                                                                     |
| ti     | BOOL    | GetCursorInfo(_Inout_ PCURSORINFO pci)                                                              |
| ti     | BOOL    | GetCursorPos(_Out_ LPPOINT lpPoint)                                                                 |
| tt     | HDC     | GetDC(_In_ HWND hwnd)                                                                               |
| ttuit  | HDC     | GetDCEX(_In_ HWND hWnd, _In_ HRGN hrgnClip, _In_ DWORD flags)                                       |
| t      | HWND    | GetDesktopWindow(void)                                                                              |
| ui     | LONG    | GetDialogBaseUnits(void)                                                                            |
| ti     | int     | GetDlgCtrlID(_In_ HWND hwndCtl)                                                                     |
| tit    | HWND    | GetDlgItem(_In_opt_ HWND hDlg, _In_ int nIDDlgItem)                                                 |
| titui  | UINT    | GetDlgItemInt(_In_ HWND hDlg, _In_ int nIDDlgItem, _Out_opt_ BOOL *lpTranslated, _In_ BOOL bSigned) |
| tisiui | UINT    | GetDlgItemText(_In_ HWND hDlg, _In_ int nIDDlgItem, _Out_ LPTSTR lpString, _In_ int nMaxCount)      |
| tiaiui | UINT    | GetDlgItemTextA(_In_ HWND hDlg, _In_ int nIDDlgItem, _Out_ LPSTR lpString, _In_ int nMaxCount)      |

|          |       |                                                                                                                                                        |
|----------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| tiwiui   | UINT  | GetDlgItemTextW(_In_ HWND hDlg, _In_ int nIDDlgItem, _Out_ LPWSTR lpString, _In_ int nMaxCount)                                                        |
| ui       | UINT  | GetDoubleClickTime(void)                                                                                                                               |
| t        | HWND  | GetFocus(void)                                                                                                                                         |
| t        | HWND  | GetForegroundWindow(void)                                                                                                                              |
| tuiittui | BOOL  | GetGestureConfig(_In_ HWND hwnd, _In_ DWORD dwReserved, _In_ DWORD dwFlags, _In_ PUINT pcIDs, _Inout_ PGESTURECONFIG pGestureConfig, _In_ UINT cbSize) |
| tuiti    | BOOL  | GetGestureExtraArgs(_In_ HGESTUREINFO hGestureInfo, _In_ UINT cbExtraArgs, _Out_ PBYTE pExtraArgs)                                                     |
| titi     | BOOL  | GetGestureInfo(_In_ HGESTUREINFO hGestureInfo, _Out_ PGESTUREINFO pGestureInfo)                                                                        |
| tuiui    | DWORD | GetGuiResources(_In_ HANDLE hProcess, _In_ DWORD uiFlags)                                                                                              |
| uiti     | BOOL  | GetGuiThreadInfo(_In_ DWORD idThread, _Inout_ LPGUITHREADINFO lpgui)                                                                                   |
| titi     | BOOL  | GetIconInfo(_In_ HICON hIcon, _Out_ PICONINFO piconinfo)                                                                                               |
| titi     | BOOL  | GetIconInfoEx(_In_ HICON hIcon, _Inout_ PICONINFOEX piconinfoex)                                                                                       |
| titi     | BOOL  | GetIconInfoExA(_In_ HICON hIcon, _Inout_ PICONINFOEX piconinfoex)                                                                                      |
| titi     | BOOL  | GetIconInfoExW(_In_ HICON hIcon, _Inout_ PICONINFOEX piconinfoex)                                                                                      |
| i        | BOOL  | GetInputState(void)                                                                                                                                    |
| ui       | UINT  | GetKBCodePage(void)                                                                                                                                    |
| uit      | HKL   | GetKeyboardLayout(_In_ DWORD idThread)                                                                                                                 |
| iti      | int   | GetKeyboardLayoutList(_In_ int nBuff, _Out_ HKL *lpList)                                                                                               |
| si       | BOOL  | GetKeyboardLayoutName(_Out_ LPTSTR pwszKLID)                                                                                                           |
| ai       | BOOL  | GetKeyboardLayoutNameA(_Out_ LPSTR pwszKLID)                                                                                                           |
| wi       | BOOL  | GetKeyboardLayoutNameW(_Out_ LPWSTR pwszKLID)                                                                                                          |
| ti       | BOOL  | GetKeyboardState(_Out_ PBYTE lpKeyState)                                                                                                               |
| ii       | int   | GetKeyboardType(_In_ int nTypeFlag)                                                                                                                    |
| uisii    | int   | GetKeyNameText(_In_ LONG lParam, _Out_ LPTSTR lpString, _In_ int cchSize)                                                                              |
| uiaii    | int   | GetKeyNameTextA(_In_ LONG lParam, _Out_ LPSTR lpString, _In_ int cchSize)                                                                              |
| uiwii    | int   | GetKeyNameTextW(_In_ LONG lParam, _Out_ LPWSTR lpString, _In_ int cchSize)                                                                             |

|          |       |                                                                                                                            |
|----------|-------|----------------------------------------------------------------------------------------------------------------------------|
| ih       | SHORT | GetKeyState(_In_ int nVirtKey)                                                                                             |
| tt       | HWND  | GetLastActivePopup(_In_ HWND hwnd)                                                                                         |
| ti       | BOOL  | GetLastInputInfo(_Out_ PLASTINPUTINFO plii)                                                                                |
| tttti    | BOOL  | GetLayeredWindowAttributes(_In_ HWND hwnd, _Out_opt_ COLORREF *pcrKey, _Out_opt_ BYTE *pbAlpha, _Out_opt_ DWORD *pdwFlags) |
| tui      | DWORD | GetListBoxInfo(_In_ HWND hwnd)                                                                                             |
| tt       | HMENU | GetMenu(_In_ HWND hwnd)                                                                                                    |
| tuiuiti  | BOOL  | GetMenuBarInfo(_In_ HWND hwnd, _In_ LONG idObject, _In_ LONG idItem, _Inout_ PMENUBARINFO pmbi)                            |
| ui       | LONG  | GetMenuCheckMarkDimensions(void)                                                                                           |
| tui      | DWORD | GetMenuContextHelpId(HMENU hmenu)                                                                                          |
| tuiuiui  | UINT  | GetMenuDefaultItem(_In_ HMENU hMenu, _In_ UINT fByPos, _In_ UINT gmdiFlags)                                                |
| tti      | BOOL  | GetMenuInfo(_In_ HMENU hmenu, _Inout_ LPMENUINFO lpcki)                                                                    |
| ti       | int   | GetMenuItemCount(_In_opt_ HMENU hMenu)                                                                                     |
| tiui     | UINT  | GetMenuItemID(_In_ HMENU hMenu, _In_ int nPos)                                                                             |
| tuiiti   | BOOL  | GetMenuItemInfo(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _Inout_ LPMENUITEMINFO lpzii)                    |
| tuiiti   | BOOL  | GetMenuItemInfoA(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _Inout_ LPMENUITEMINFO lpzii)                   |
| tuiiti   | BOOL  | GetMenuItemInfoW(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _Inout_ LPMENUITEMINFO lpzii)                   |
| ttuiti   | BOOL  | GetMenuItemRect(_In_opt_ HWND hWnd, _In_ HMENU hMenu, _In_ UINT uItem, _Out_ LPRECT lprci)                                 |
| tuiuiui  | UINT  | GetMenuState(_In_ HMENU hMenu, _In_ UINT uId, _In_ UINT uFlags)                                                            |
| tuisiuii | int   | GetMenuString(_In_ HMENU hMenu, _In_ UINT uIDItem, _Out_opt_ LPTSTR lpString, _In_ int nMaxCount, _In_ UINT uFlag)         |
| tuiaiuii | int   | GetMenuStringA(_In_ HMENU hMenu, _In_ UINT uIDItem, _Out_opt_ LPSTR lpString, _In_ int nMaxCount, _In_ UINT uFlag)         |
| tuiwiuii | int   | GetMenuStringW(_In_ HMENU hMenu, _In_ UINT uIDItem, _Out_opt_ LPWSTR lpString, _In_ int nMaxCount, _In_ UINT uFlag)        |
| ttuiuii  | BOOL  | GetMessage(_Out_ LPMSG lpMsg, _In_opt_ HWND hWnd, _In_ UINT wParamFilterMin, _In_ UINT wParamFilterMax)                    |
| ttuiuii  | BOOL  | GetMessageA(_Out_ LPMSG lpMsg, _In_opt_ HWND hWnd,                                                                         |

|           |        |                                                                                                                                                |
|-----------|--------|------------------------------------------------------------------------------------------------------------------------------------------------|
|           |        | _In_ UINT wParamFilterMin, _In_ UINT wParamFilterMax)                                                                                          |
| t         | LPARAM | GetMessageExtraInfo(void)                                                                                                                      |
| ui        | DWORD  | GetMessagePos(void)                                                                                                                            |
| ui        | LONG   | GetMessageTime(void)                                                                                                                           |
| ttuiiii   | BOOL   | GetMessageW(_Out_ LPMSG lpMsg, _In_opt_ HWND hWnd, _In_ UINT wParamFilterMin, _In_ UINT wParamFilterMax)                                       |
| tii       | BOOL   | GetMonitorInfo(_In_ HMONITOR hMonitor, _Out_ LPMONITORINFO lpmi)                                                                               |
| tii       | BOOL   | GetMonitorInfoA(_In_ HMONITOR hMonitor, _Out_ LPMONITORINFO lpmi)                                                                              |
| tii       | BOOL   | GetMonitorInfoW(_In_ HMONITOR hMonitor, _Out_ LPMONITORINFO lpmi)                                                                              |
| uittuiii  | int    | GetMouseMovePointsEx(_In_ UINT cbSize, _In_ LPMOUSEMOVEPOINT lppt, _Out_ LPMOUSEMOVEPOINT lpptBuf, _In_ int nBufPoints, _In_ DWORD resolution) |
| ttit      | HWND   | GetNextDlgGroupItem(_In_ HWND hDlg, _In_opt_ HWND hCtl, _In_ BOOL bPrevious)                                                                   |
| ttit      | HWND   | GetNextDlgTabItem(_In_ HWND hDlg, _In_opt_ HWND hCtl, _In_ BOOL bPrevious)                                                                     |
| t         | HWND   | GetOpenClipboardWindow(void)                                                                                                                   |
| tt        | HWND   | GetParent(_In_ HWND hWnd)                                                                                                                      |
| ti        | BOOL   | GetPhysicalCursorPos(_Out_ LPPOINT lpPoint)                                                                                                    |
| tii       | int    | GetPriorityClipboardFormat(_In_ UINT *paFormatPriorityList, _In_ int cFormats)                                                                 |
| ti        | BOOL   | GetProcessDefaultLayout(_Out_ DWORD *pdwDefaultLayout)                                                                                         |
| t         | HWND   | GetProcessWindowStation(void)                                                                                                                  |
| tst       | HANDLE | GetProp(_In_ HWND hWnd, _In_ LPCTSTR lpString)                                                                                                 |
| tat       | HANDLE | GetPropA(_In_ HWND hWnd, _In_ LPCSTR lpString)                                                                                                 |
| twt       | HANDLE | GetPropW(_In_ HWND hWnd, _In_ LPCWSTR lpString)                                                                                                |
| uiui      | DWORD  | GetQueueStatus(_In_ UINT flags)                                                                                                                |
| ttuiui    | UINT   | GetRawInputBuffer(_Out_opt_ PRAWINPUT pData, _Inout_ PUINT pcbSize, _In_ UINT cbSizeHeader)                                                    |
| tuittuiui | UINT   | GetRawInputData(_In_ HRAWINPUT hRawInput, _In_ UINT uiCommand, _Out_opt_ LPVOID pData, _Inout_ PUINT pcbSize, _In_ UINT cbSizeHeader)          |
| tuittui   | UINT   | GetRawInputDeviceInfo(_In_opt_ HANDLE hDevice, _In_ UINT uiCommand, _Inout_opt_ LPVOID pData, _Inout_ PUINT pcbSize)                           |

|         |        |                                                                                                                                          |
|---------|--------|------------------------------------------------------------------------------------------------------------------------------------------|
| tuittui | UINT   | GetRawInputDeviceInfoA(_In_opt_ HANDLE hDevice, _In_ UINT uiCommand, _Inout_opt_ LPVOID pData, _Inout_ PUINT pcbSize)                    |
| tuittui | UINT   | GetRawInputDeviceInfoW(_In_opt_ HANDLE hDevice, _In_ UINT uiCommand, _Inout_opt_ LPVOID pData, _Inout_ PUINT pcbSize)                    |
| ttuiui  | UINT   | GetRawInputDeviceList(_Out_opt_ PRAWINPUTDEVICELIST pRawInputDeviceList, _Inout_ PUINT puiNumDevices, _In_ UINT cbSize)                  |
| ttuiui  | UINT   | GetRegisteredRawInputDevices(_Out_opt_ PRAWINPUTDEVICE pRawInputDevices, _Inout_ PUINT puiNumDevices, _In_ UINT cbSize)                  |
| tuiti   | BOOL   | GetScrollBarInfo(_In_ HWND hwnd, _In_ LONG idObject, _Out_ PSCROLLBARINFO psbi)                                                          |
| titi    | BOOL   | GetScrollInfo(_In_ HWND hwnd, _In_ int fnBar, _Inout_ LPSCROLLINFO lpsi)                                                                 |
| tii     | int    | GetScrollPos(_In_ HWND hWnd, _In_ int nBar)                                                                                              |
| titti   | BOOL   | GetScrollRange(_In_ HWND hWnd, _In_ int nBar, _Out_ LPINT lpMinPos, _Out_ LPINT lpMaxPos)                                                |
| t       | HWND   | GetShellWindow(void)                                                                                                                     |
| tit     | HMENU  | GetSubMenu(_In_ HMENU hMenu, _In_ int nPos)                                                                                              |
| iui     | DWORD  | GetSysColor(_In_ int nIndex)                                                                                                             |
| it      | HBRUSH | GetSysColorBrush(_In_ int nIndex)                                                                                                        |
| tit     | HMENU  | GetSystemMenu(_In_ HWND hWnd, _In_ BOOL bRevert)                                                                                         |
| ii      | int    | GetSystemMetrics(_In_ int nIndex)                                                                                                        |
| tsiitui | DWORD  | GetTabbedTextExtent(_In_ HDC hDC, _In_ LPCTSTR lpString, _In_ int nCount, _In_ int nTabPositions, _In_ const LPINT lpnTabStopPositions)  |
| taiitui | DWORD  | GetTabbedTextExtentA(_In_ HDC hDC, _In_ LPCSTR lpString, _In_ int nCount, _In_ int nTabPositions, _In_ const LPINT lpnTabStopPositions)  |
| twiitui | DWORD  | GetTabbedTextExtentW(_In_ HDC hDC, _In_ LPCWSTR lpString, _In_ int nCount, _In_ int nTabPositions, _In_ const LPINT lpnTabStopPositions) |
| uit     | HDESK  | GetThreadDesktop(_In_ DWORD dwThreadId)                                                                                                  |
| titi    | BOOL   | GetTitleBarInfo(_In_ HWND hwnd, _Inout_ PTITLEBARINFO pti)                                                                               |
| tt      | HWND   | GetTopWindow(_In_opt_ HWND hWnd)                                                                                                         |
| tuitii  | BOOL   | GetTouchInputInfo(_In_ HTOUCHINPUT hTouchInput, _In_ UINT cInputs, _Out_ PTOUCHINPUT pInputs, _In_ int cbSize)                           |

|         |       |                                                                                                                                                                       |
|---------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiti   | BOOL  | GetUpdatedClipboardFormats(_Out_ PUINT lpuiFormats, _In_ UINT cFormats, _Out_ PUINT pcFormatsOut)                                                                     |
| ttii    | BOOL  | GetUpdateRect(_In_ HWND hWnd, _Out_ LPRECT lpRect, _In_ BOOL bErase)                                                                                                  |
| ttii    | int   | GetUpdateRgn(_In_ HWND hWnd, _In_ HRGN hRgn, _In_ BOOL bErase)                                                                                                        |
| tituiti | BOOL  | GetObjectInformation(_In_ HANDLE hObj, _In_ int nIndex, _Out_opt_ PVOID pvInfo, _In_ DWORD nLength, _Out_opt_ LPDWORD lpnLengthNeeded)                                |
| tituiti | BOOL  | GetObjectInformationA(_In_ HANDLE hObj, _In_ int nIndex, _Out_opt_ PVOID pvInfo, _In_ DWORD nLength, _Out_opt_ LPDWORD lpnLengthNeeded)                               |
| tituiti | BOOL  | GetObjectInformationW(_In_ HANDLE hObj, _In_ int nIndex, _Out_opt_ PVOID pvInfo, _In_ DWORD nLength, _Out_opt_ LPDWORD lpnLengthNeeded)                               |
| tttuiti | BOOL  | GetObjectSecurity(_In_ HANDLE hObj, _In_ PSECURITY_INFORMATION pSIRequested, _Inout_opt_ PSECURITY_DESCRIPTOR pSD, _In_ DWORD nLength, _Out_ LPDWORD lpnLengthNeeded) |
| tuit    | HWND  | GetWindow(_In_ HWND hWnd, _In_ UINT uCmd)                                                                                                                             |
| tui     | DWORD | GetWindowContextHelpId(HWND hwnd)                                                                                                                                     |
| tt      | HDC   | GetWindowDC(_In_ HWND hwnd)                                                                                                                                           |
| tti     | BOOL  | GetWindowDisplayAffinity(_In_ HWND hWnd, _Out_ DWORD *dwAffinity)                                                                                                     |
| tti     | BOOL  | GetWindowInfo(_In_ HWND hwnd, _Inout_ WINDOWINFO pwi)                                                                                                                 |
| tiui    | LONG  | GetWindowLong(_In_ HWND hWnd, _In_ int nIndex)                                                                                                                        |
| tiui    | LONG  | GetWindowLongA(_In_ HWND hWnd, _In_ int nIndex)                                                                                                                       |
| tiui    | LONG  | GetWindowLongW(_In_ HWND hWnd, _In_ int nIndex)                                                                                                                       |
| tsuiui  | UINT  | GetWindowModuleFileName(_In_ HWND hwnd, _Out_ LPTSTR lpszFileName, _In_ UINT cchFileNameMax)                                                                          |
| tauuiui | UINT  | GetWindowModuleFileNameA(_In_ HWND hwnd, _Out_ LPSTR lpszFileName, _In_ UINT cchFileNameMax)                                                                          |
| twuiui  | UINT  | GetWindowModuleFileNameW(_In_ HWND hwnd, _Out_ LPWSTR lpszFileName, _In_ UINT cchFileNameMax)                                                                         |
| tti     | BOOL  | GetWindowPlacement(_In_ HWND hWnd, _Inout_ WINDOWPLACEMENT *lpwndpl)                                                                                                  |
| tti     | BOOL  | GetWindowRect(_In_ HWND hWnd, _Out_ LPRECT lpRect)                                                                                                                    |
| tti     | int   | GetWindowRgn(_In_ HWND hWnd, _In_ HRGN hRgn)                                                                                                                          |
| tti     | int   | GetWindowRgnBox(_In_ HWND hWnd, _Out_ LPRECT lprc)                                                                                                                    |

|         |       |                                                                                                                                                                                 |
|---------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsii    | int   | GetWindowText(_In_ HWND hWnd, _Out_ LPCTSTR lpString, _In_ int nMaxCount)                                                                                                       |
| taii    | int   | GetWindowTextA(_In_ HWND hWnd, _Out_ LPSTR lpString, _In_ int nMaxCount)                                                                                                        |
| ti      | int   | GetWindowTextLength(_In_ HWND hWnd)                                                                                                                                             |
| ti      | int   | GetWindowTextLengthA(_In_ HWND hWnd)                                                                                                                                            |
| ti      | int   | GetWindowTextLengthW(_In_ HWND hWnd)                                                                                                                                            |
| twii    | int   | GetWindowTextW(_In_ HWND hWnd, _Out_ LPWSTR lpString, _In_ int nMaxCount)                                                                                                       |
| ttui    | DWORD | GetWindowThreadProcessId(_In_ HWND hWnd, _Out_opt_ LPDWORD lpdwProcessId)                                                                                                       |
| tttiii  | BOOL  | GrayString(_In_ HDC hDC, _In_ HBRUSH hBrush, _In_ GRAYSTRINGPROC lpOutputFunc, _In_ LPARAM lpData, _In_ int nCount, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight)  |
| tttiii  | BOOL  | GrayStringA(_In_ HDC hDC, _In_ HBRUSH hBrush, _In_ GRAYSTRINGPROC lpOutputFunc, _In_ LPARAM lpData, _In_ int nCount, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight) |
| tttiii  | BOOL  | GrayStringW(_In_ HDC hDC, _In_ HBRUSH hBrush, _In_ GRAYSTRINGPROC lpOutputFunc, _In_ LPARAM lpData, _In_ int nCount, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight) |
| ti      | BOOL  | HideCaret(_In_opt_ HWND hWnd)                                                                                                                                                   |
| ttuiii  | BOOL  | HiliteMenuItem(_In_ HWND hWnd, _In_ HMENU hMenu, _In_ UINT uItemHilite, _In_ UINT uHilite)                                                                                      |
| tii     | BOOL  | ImpersonateDdeClientWindow(_In_ HWND hWndClient, _In_ HWND hWndServer)                                                                                                          |
| tiii    | BOOL  | InflateRect(_Inout_ LPRECT lprc, _In_ int dx, _In_ int dy)                                                                                                                      |
| i       | BOOL  | InSendMessage(void)                                                                                                                                                             |
| tui     | DWORD | InSendMessageEx(_Reserved_ LPVOID lpReserved)                                                                                                                                   |
| tuiitsi | BOOL  | InsertMenu(_In_ HMENU hMenu, _In_ UINT uPosition, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCTSTR lpNewItem)                                                       |
| tuiitai | BOOL  | InsertMenuA(_In_ HMENU hMenu, _In_ UINT uPosition, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCSTR lpNewItem)                                                       |
| tuiiti  | BOOL  | InsertMenuItem(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _In_ LPCMENUIITEMINFO lpmi)                                                                            |
| tuiiti  | BOOL  | InsertMenuItemA(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _In_ LPCMENUIITEMINFO lpmi)                                                                           |
|         |       | InsertMenuItemW(_In_ HMENU hMenu, _In_ UINT uItem,                                                                                                                              |

|          |      |                                                                                                                            |
|----------|------|----------------------------------------------------------------------------------------------------------------------------|
| tuiiti   | BOOL | _In_ BOOL fByPosition, _In_ LPCMENUIITEMINFO lpmii)                                                                        |
| tuiuitwi | BOOL | InsertMenuW(_In_ HMENU hMenu, _In_ UINT uPosition, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCWSTR lpNewItem) |
| twii     | int  | InternalGetWindowText(_In_ HWND hWnd, _Out_ LPWSTR lpString, _In_ int nMaxCount)                                           |
| ttti     | BOOL | IntersectRect(_Out_ LPRECT lprcDst, _In_ const RECT *lprcSrc1, _In_ const RECT *lprcSrc2)                                  |
| ttii     | BOOL | InvalidateRect(_In_ HWND hWnd, _In_ const RECT *lpRect, _In_ BOOL bErase)                                                  |
| ttiii    | BOOL | InvalidateRgn(_In_ HWND hWnd, _In_ HRGN hRgn, _In_ BOOL bErase)                                                            |
| ttti     | BOOL | InvertRect(_In_ HDC hDC, _In_ const RECT *lprc)                                                                            |
| yi       | BOOL | IsCharAlpha(_In_ TCHAR ch)                                                                                                 |
| yi       | BOOL | IsCharAlphaA(_In_ TCHAR ch)                                                                                                |
| yi       | BOOL | IsCharAlphaNumeric(_In_ TCHAR ch)                                                                                          |
| yi       | BOOL | IsCharAlphaNumericA(_In_ TCHAR ch)                                                                                         |
| yi       | BOOL | IsCharAlphaNumericW(_In_ TCHAR ch)                                                                                         |
| yi       | BOOL | IsCharAlphaW(_In_ TCHAR ch)                                                                                                |
| yi       | BOOL | IsCharLower(_In_ TCHAR ch)                                                                                                 |
| yi       | BOOL | IsCharLowerA(_In_ TCHAR ch)                                                                                                |
| yi       | BOOL | IsCharLowerW(_In_ TCHAR ch)                                                                                                |
| yi       | BOOL | IsCharUpper(_In_ TCHAR ch)                                                                                                 |
| yi       | BOOL | IsCharUpperA(_In_ TCHAR ch)                                                                                                |
| yi       | BOOL | IsCharUpperW(_In_ TCHAR ch)                                                                                                |
| ttti     | BOOL | IsChild(_In_ HWND hWndParent, _In_ HWND hWnd)                                                                              |
| uii      | BOOL | IsClipboardFormatAvailable(_In_ UINT format)                                                                               |
| ttti     | BOOL | IsDialogMessage(_In_ HWND hDlg, _In_ LPMSG lpMsg)                                                                          |
| ttti     | BOOL | IsDialogMessageA(_In_ HWND hDlg, _In_ LPMSG lpMsg)                                                                         |
| ttti     | BOOL | IsDialogMessageW(_In_ HWND hDlg, _In_ LPMSG lpMsg)                                                                         |
| tiui     | UINT | IsDlgButtonChecked(_In_ HWND hDlg, _In_ int nIDButton)                                                                     |
| ii       | BOOL | IsGUIThread(_In_ BOOL bConvert)                                                                                            |
| ti       | BOOL | IsHungAppWindow(_In_ HWND hwnd)                                                                                            |
| ti       | BOOL | IsIconic(_In_ HWND hwnd)                                                                                                   |
| ti       | BOOL | IsMenu(_In_ HMENU hMenu)                                                                                                   |
| i        | BOOL | IsProcessDPIAware(void)                                                                                                    |

|          |         |                                                                                             |
|----------|---------|---------------------------------------------------------------------------------------------|
| ti       | BOOL    | IsRectEmpty(_In_ const RECT *lprc)                                                          |
| tii      | BOOL    | IsTouchWindow(_In_ HWND hWnd, _Out_opt_ PULONG pulFlags)                                    |
| ti       | BOOL    | IsWindow(_In_opt_ HWND hWnd)                                                                |
| ti       | BOOL    | IsWindowEnabled(_In_ HWND hWnd)                                                             |
| ti       | BOOL    | IsWindowRedirectedForPrint(_In_ HWND hWnd)                                                  |
| ti       | BOOL    | IsWindowUnicode(_In_ HWND hWnd)                                                             |
| ti       | BOOL    | IsWindowVisible(_In_ HWND hWnd)                                                             |
| uii      | BOOL    | IsWinEventHookInstalled(_In_ DWORD event)                                                   |
| i        | BOOL    | IsWow64Message(void)                                                                        |
| ti       | BOOL    | IsZoomed(_In_ HWND hWnd)                                                                    |
| ucucuiti | VOID    | keybd_event(_In_ BYTE bVk, _In_ BYTE bScan, _In_ DWORD dwFlags, _In_ ULONG_PTR dwExtraInfo) |
| tii      | BOOL    | KillTimer(_In_opt_ HWND hWnd, _In_ UINT_PTR uIDEvent)                                       |
| tst      | HACCEL  | LoadAccelerator(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpTableName)                     |
| tat      | HACCEL  | LoadAcceleratorsA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpTableName)                    |
| twt      | HACCEL  | LoadAcceleratorsW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpTableName)                   |
| tst      | HBITMAP | LoadBitmap(_In_ HINSTANCE hInstance, _In_ LPCTSTR lpBitmapName)                             |
| tat      | HBITMAP | LoadBitmapA(_In_ HINSTANCE hInstance, _In_ LPCSTR lpBitmapName)                             |
| twt      | HBITMAP | LoadBitmapW(_In_ HINSTANCE hInstance, _In_ LPCWSTR lpBitmapName)                            |
| tst      | HCURSOR | LoadCursor(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpCursorName)                         |
| tat      | HCURSOR | LoadCursorA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpCursorName)                         |
| st       | HCURSOR | LoadCursorFromFile(_In_ LPCTSTR lpFileName)                                                 |
| at       | HCURSOR | LoadCursorFromFileA(_In_ LPCSTR lpFileName)                                                 |
| wt       | HCURSOR | LoadCursorFromFileW(_In_ LPCWSTR lpFileName)                                                |
| twt      | HCURSOR | LoadCursorW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpCursorName)                        |
| tst      | HICON   | LoadIcon(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpIconName)                             |
| tat      | HICON   | LoadIconA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpIconName)                             |

|          |        |                                                                                                                                        |
|----------|--------|----------------------------------------------------------------------------------------------------------------------------------------|
|          |        | lpIconName)                                                                                                                            |
| twt      | HICON  | LoadIconW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpIconName)                                                                       |
| tsuiiuit | HANDLE | LoadImage(_In_opt_ HINSTANCE hinst, _In_ LPCTSTR lpszName, _In_ UINT uType, _In_ int cxDesired, _In_ int cyDesired, _In_ UINT fuLoad)  |
| tauiiuit | HANDLE | LoadImageA(_In_opt_ HINSTANCE hinst, _In_ LPCSTR lpszName, _In_ UINT uType, _In_ int cxDesired, _In_ int cyDesired, _In_ UINT fuLoad)  |
| twuiiuit | HANDLE | LoadImageW(_In_opt_ HINSTANCE hinst, _In_ LPCWSTR lpszName, _In_ UINT uType, _In_ int cxDesired, _In_ int cyDesired, _In_ UINT fuLoad) |
| suit     | HKL    | LoadKeyboardLayout(_In_ LPCTSTR pwszKLID, _In_ UINT Flags)                                                                             |
| auit     | HKL    | LoadKeyboardLayoutA(_In_ LPCSTR pwszKLID, _In_ UINT Flags)                                                                             |
| wuit     | HKL    | LoadKeyboardLayoutW(_In_ LPCWSTR pwszKLID, _In_ UINT Flags)                                                                            |
| tst      | HMENU  | LoadMenu(_In_opt_ HINSTANCE hInstance, _In_ LPCTSTR lpMenuName)                                                                        |
| tat      | HMENU  | LoadMenuA(_In_opt_ HINSTANCE hInstance, _In_ LPCSTR lpMenuName)                                                                        |
| tt       | HMENU  | LoadMenuIndirect(_In_ const MENUTEMPLATE *lpMenuTemplate)                                                                              |
| tt       | HMENU  | LoadMenuIndirectA(_In_ const MENUTEMPLATE *lpMenuTemplate)                                                                             |
| tt       | HMENU  | LoadMenuIndirectW(_In_ const MENUTEMPLATE *lpMenuTemplate)                                                                             |
| twt      | HMENU  | LoadMenuW(_In_opt_ HINSTANCE hInstance, _In_ LPCWSTR lpMenuName)                                                                       |
| tuisii   | int    | LoadString(_In_opt_ HINSTANCE hInstance, _In_ UINT uID, _Out_ LPTSTR lpBuffer, _In_ int nBufferMax)                                    |
| tuiiai   | int    | LoadStringA(_In_opt_ HINSTANCE hInstance, _In_ UINT uID, _Out_ LPSTR lpBuffer, _In_ int nBufferMax)                                    |
| tuiwii   | int    | LoadStringW(_In_opt_ HINSTANCE hInstance, _In_ UINT uID, _Out_ LPWSTR lpBuffer, _In_ int nBufferMax)                                   |
| uii      | BOOL   | LockSetForegroundWindow(_In_ UINT uLockCode)                                                                                           |
| ti       | BOOL   | LockWindowUpdate(_In_ HWND hWndLock)                                                                                                   |
| i        | BOOL   | LockWorkStation(void)                                                                                                                  |
| tti      | BOOL   | LogicalToPhysicalPoint(_In_ HWND hWnd, _Inout_ LPPOINT lpPoint)                                                                        |

|          |      |                                                                                                                                |
|----------|------|--------------------------------------------------------------------------------------------------------------------------------|
| tii      | int  | LookupIconIdFromDirectory(_In_ PBYTE presbits, _In_ BOOL fIcon)                                                                |
| tiiiii   | int  | LookupIconIdFromDirectoryEx(_In_ PBYTE presbits, _In_ BOOL fIcon, _In_ int cxDesired, _In_ int cyDesired, _In_ UINT Flags)     |
| tii      | BOOL | MapDialogRect(_In_ HWND hDlg, _Inout_ LPRECT lpRect)                                                                           |
| uiuiui   | UINT | MapVirtualKey(_In_ UINT uCode, _In_ UINT uMapType)                                                                             |
| uiuiui   | UINT | MapVirtualKeyA(_In_ UINT uCode, _In_ UINT uMapType)                                                                            |
| uiuitui  | UINT | MapVirtualKeyEx(_In_ UINT uCode, _In_ UINT uMapType, _Inout_opt_ HKL dwhkl)                                                    |
| uiuitui  | UINT | MapVirtualKeyExA(_In_ UINT uCode, _In_ UINT uMapType, _Inout_opt_ HKL dwhkl)                                                   |
| uiuitui  | UINT | MapVirtualKeyExW(_In_ UINT uCode, _In_ UINT uMapType, _Inout_opt_ HKL dwhkl)                                                   |
| uiuiui   | UINT | MapVirtualKeyW(_In_ UINT uCode, _In_ UINT uMapType)                                                                            |
| ttuii    | int  | MapWindowPoints(_In_ HWND hWndFrom, _In_ HWND hWndTo, _Inout_ LPPOINT lpPoints, _In_ UINT cPoints)                             |
| tii6i    | int  | MenuItemFromPoint(_In_opt_ HWND hWnd, _In_ HMENU hMenu, _In_ POINT ptScreen)                                                   |
| uii      | BOOL | MessageBeep(_In_ UINT uType)                                                                                                   |
| tssuii   | int  | MessageBox(_In_opt_ HWND hWnd, _In_opt_ LPCTSTR lpText, _In_opt_ LPCTSTR lpCaption, _In_ UINT uType)                           |
| tauii    | int  | MessageBoxA(_In_opt_ HWND hWnd, _In_opt_ LPCSTR lpText, _In_opt_ LPCSTR lpCaption, _In_ UINT uType)                            |
| tssuiuhi | int  | MessageBoxEx(_In_opt_ HWND hWnd, _In_opt_ LPCTSTR lpText, _In_opt_ LPCTSTR lpCaption, _In_ UINT uType, _In_ WORD wLanguageId)  |
| tauiuhi  | int  | MessageBoxExA(_In_opt_ HWND hWnd, _In_opt_ LPCSTR lpText, _In_opt_ LPCSTR lpCaption, _In_ UINT uType, _In_ WORD wLanguageId)   |
| twwuiuhi | int  | MessageBoxExW(_In_opt_ HWND hWnd, _In_opt_ LPCWSTR lpText, _In_opt_ LPCWSTR lpCaption, _In_ UINT uType, _In_ WORD wLanguageId) |
| ti       | int  | MessageBoxIndirect(_In_ const LPMSGBOXPARAMS lpMsgBoxParams)                                                                   |
| ti       | int  | MessageBoxIndirectA(_In_ const LPMSGBOXPARAMS lpMsgBoxParams)                                                                  |
| ti       | int  | MessageBoxIndirectW(_In_ const LPMSGBOXPARAMS lpMsgBoxParams)                                                                  |
| twwuii   | int  | MessageBoxW(_In_opt_ HWND hWnd, _In_opt_ LPCWSTR lpText, _In_opt_ LPCWSTR lpCaption, _In_ UINT uType)                          |

|             |          |                                                                                                                                                   |
|-------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiuitsi    | BOOL     | ModifyMenu(_In_ HMENU hMnu, _In_ UINT uPosition, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCTSTR lpNewItem)                          |
| tuiuitai    | BOOL     | ModifyMenuA(_In_ HMENU hMnu, _In_ UINT uPosition, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCSTR lpNewItem)                          |
| tuiuitwi    | BOOL     | ModifyMenuW(_In_ HMENU hMnu, _In_ UINT uPosition, _In_ UINT uFlags, _In_ UINT_PTR uIDNewItem, _In_opt_ LPCWSTR lpNewItem)                         |
| i6uit       | HMONITOR | MonitorFromPoint(_In_ POINT pt, _In_ DWORD dwFlags)                                                                                               |
| tuit        | HMONITOR | MonitorFromRect(_In_ LPCRECT lprc, _In_ DWORD dwFlags)                                                                                            |
| tuit        | HMONITOR | MonitorFromWindow(_In_ HWND hwnd, _In_ DWORD dwFlags)                                                                                             |
| uiuiuiuiti  | VOID     | mouse_event(_In_ DWORD dwFlags, _In_ DWORD dx, _In_ DWORD dy, _In_ DWORD dwData, _In_ ULONG_PTR dwExtraInfo)                                      |
| tiiiiii     | BOOL     | MoveWindow(_In_ HWND hWnd, _In_ int X, _In_ int Y, _In_ int nWidth, _In_ int nHeight, _In_ BOOL bRepaint)                                         |
| uitiuiuiui  | DWORD    | MsgWaitForMultipleObjects(_In_ DWORD nCount, _In_ const HANDLE *pHandles, _In_ BOOL bWaitAll, _In_ DWORD dwMilliseconds, _In_ DWORD dwWakeMask)   |
| uitiuiuiuii | DWORD    | MsgWaitForMultipleObjectsEx(_In_ DWORD nCount, _In_ const HANDLE *pHandles, _In_ DWORD dwMilliseconds, _In_ DWORD dwWakeMask, _In_ DWORD dwFlags) |
| uitiuiui    | VOID     | NotifyWinEvent(_In_ DWORD event, _In_ HWND hwnd, _In_ LONG idObject, _In_ LONG idChild)                                                           |
| uhui        | DWORD    | OemKeyScan(_In_ WORD wOemChar)                                                                                                                    |
| asi         | BOOL     | OemToChar(_In_ LPCSTR lpszSrc, _Out_ LPTSTR lpszDst)                                                                                              |
| aaI         | BOOL     | OemToCharA(_In_ LPCSTR lpszSrc, _Out_ LPSTR lpszDst)                                                                                              |
| asuui       | BOOL     | OemToCharBuff(_In_ LPCSTR lpszSrc, _Out_ LPTSTR lpszDst, _In_ DWORD cchDstLength)                                                                 |
| aauii       | BOOL     | OemToCharBuffA(_In_ LPCSTR lpszSrc, _Out_ LPSTR lpszDst, _In_ DWORD cchDstLength)                                                                 |
| awuui       | BOOL     | OemToCharBuffW(_In_ LPCSTR lpszSrc, _Out_ LPWSTR lpszDst, _In_ DWORD cchDstLength)                                                                |
| awi         | BOOL     | OemToCharW(_In_ LPCSTR lpszSrc, _Out_ LPWSTR lpszDst)                                                                                             |
| tiii        | BOOL     | OffsetRect(_Inout_ LPRECT lprc, _In_ int dx, _In_ int dy)                                                                                         |
| ti          | BOOL     | OpenClipboard(_In_opt_ HWND hWndNewOwner)                                                                                                         |
| suiiuit     | HDESK    | OpenDesktop(_In_ LPTSTR lpszDesktop, _In_ DWORD dwFlags, _In_ BOOL fInherit, _In_ ACCESS_MASK                                                     |

|          |         |                                                                                                                                      |
|----------|---------|--------------------------------------------------------------------------------------------------------------------------------------|
|          |         | dwDesiredAccess)                                                                                                                     |
| auuiuit  | HDESK   | OpenDesktopA(_In_ LPSTR lpszDesktop, _In_ DWORD dwFlags, _In_ BOOL fInherit, _In_ ACCESS_MASK dwDesiredAccess)                       |
| wuiiuit  | HDESK   | OpenDesktopW(_In_ LPWSTR lpszDesktop, _In_ DWORD dwFlags, _In_ BOOL fInherit, _In_ ACCESS_MASK dwDesiredAccess)                      |
| ti       | BOOL    | OpenIcon(_In_ HWND hwnd)                                                                                                             |
| uiiuit   | HDESK   | OpenInputDesktop(_In_ DWORD dwFlags, _In_ BOOL fInherit, _In_ ACCESS_MASK dwDesiredAccess)                                           |
| siuit    | HWINSTA | OpenWindowStation(_In_ LPTSTR lpszWinSta, _In_ BOOL fInherit, _In_ ACCESS_MASK dwDesiredAccess)                                      |
| aiuit    | HWINSTA | OpenWindowStationA(_In_ LPSTR lpszWinSta, _In_ BOOL fInherit, _In_ ACCESS_MASK dwDesiredAccess)                                      |
| wiuit    | HWINSTA | OpenWindowStationW(_In_ LPWSTR lpszWinSta, _In_ BOOL fInherit, _In_ ACCESS_MASK dwDesiredAccess)                                     |
| uittt    | LPARAM  | PackDDEIParam(_In_ UINT msg, _In_ UINT_PTR uiLo, _In_ UINT_PTR uiHi)                                                                 |
| ti       | BOOL    | PaintDesktop(_In_ HDC hdc)                                                                                                           |
| ttuiuiui | BOOL    | PeekMessage(_Out_ LPMSG lpMsg, _In_opt_ HWND hWnd, _In_ UINT wParamFilterMin, _In_ UINT wParamFilterMax, _In_ UINT wParamRemoveMsg)  |
| ttuiuiui | BOOL    | PeekMessageA(_Out_ LPMSG lpMsg, _In_opt_ HWND hWnd, _In_ UINT wParamFilterMin, _In_ UINT wParamFilterMax, _In_ UINT wParamRemoveMsg) |
| ttuiuiui | BOOL    | PeekMessageW(_Out_ LPMSG lpMsg, _In_opt_ HWND hWnd, _In_ UINT wParamFilterMin, _In_ UINT wParamFilterMax, _In_ UINT wParamRemoveMsg) |
| tti      | BOOL    | PhysicalToLogicalPoint(_In_ HWND hWnd, _Inout_ LPPOINT lpPoint)                                                                      |
| tuitti   | BOOL    | PostMessage(_In_opt_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                               |
| tuitti   | BOOL    | PostMessageA(_In_opt_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                              |
| tuitti   | BOOL    | PostMessageW(_In_opt_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                              |
| ii       | VOID    | PostQuitMessage(_In_ int nExitCode)                                                                                                  |
| uiiuitti | BOOL    | PostThreadMessage(_In_ DWORD idThread, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                        |
| uiiuitti | BOOL    | PostThreadMessageA(_In_ DWORD idThread, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                       |

|             |            |                                                                                                                                                                                         |
|-------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuitti     | BOOL       | PostThreadMessageW(_In_ DWORD idThread, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                          |
| ttuii       | BOOL       | PrintWindow(HWND hwnd, HDC hdcBlt, UINT nFlags)                                                                                                                                         |
| siittuiuiui | UINT       | PrivateExtractIcons(_In_ LPCTSTR lpszFile, _In_ int nIconIndex, _In_ int cxIcon, _In_ int cyIcon, _Out_opt_ HICON *phicon, _Out_opt_ UINT *piconid, _In_ UINT nIcons, _In_ UINT flags)  |
| aiittuiuiui | UINT       | PrivateExtractIconsA(_In_ LPCSTR lpszFile, _In_ int nIconIndex, _In_ int cxIcon, _In_ int cyIcon, _Out_opt_ HICON *phicon, _Out_opt_ UINT *piconid, _In_ UINT nIcons, _In_ UINT flags)  |
| wiittuiuiui | UINT       | PrivateExtractIconsW(_In_ LPCWSTR lpszFile, _In_ int nIconIndex, _In_ int cxIcon, _In_ int cyIcon, _Out_opt_ HICON *phicon, _Out_opt_ UINT *piconid, _In_ UINT nIcons, _In_ UINT flags) |
| ti6i        | BOOL       | PtInRect(_In_ const RECT *lprc, _In_ POINT pt)                                                                                                                                          |
| ti6t        | HWND       | RealChildWindowFromPoint(_In_ HWND hwndParent, _In_ POINT ptParentClientCoords)                                                                                                         |
| tsuiui      | UINT       | RealGetWindowClass(_In_ HWND hwnd, _Out_ LPTSTR pszType, _In_ UINT cchType)                                                                                                             |
| tauiui      | UINT       | RealGetWindowClassA(_In_ HWND hwnd, _Out_ LPSTR pszType, _In_ UINT cchType)                                                                                                             |
| twuiui      | UINT       | RealGetWindowClassW(_In_ HWND hwnd, _Out_ LPWSTR pszType, _In_ UINT cchType)                                                                                                            |
| ttuii       | BOOL       | RedrawWindow(_In_ HWND hwnd, _In_ const RECT *lprcUpdate, _In_ HRGN hrgnUpdate, _In_ UINT flags)                                                                                        |
| tuh         | ATOM       | RegisterClass(_In_ const WNDCLASS *lpWndClass)                                                                                                                                          |
| tuh         | ATOM       | RegisterClassA(_In_ const WNDCLASS *lpWndClass)                                                                                                                                         |
| tuh         | ATOM       | RegisterClassEx(_In_ const WNDCLASSEX *lpwcx)                                                                                                                                           |
| tuh         | ATOM       | RegisterClassExA(_In_ const WNDCLASSEX *lpwcx)                                                                                                                                          |
| tuh         | ATOM       | RegisterClassExW(_In_ const WNDCLASSEX *lpwcx)                                                                                                                                          |
| tuh         | ATOM       | RegisterClassW(_In_ const WNDCLASS *lpWndClass)                                                                                                                                         |
| sui         | UINT       | RegisterClipboardFormat(_In_ LPCTSTR lpszFormat)                                                                                                                                        |
| au          | UINT       | RegisterClipboardFormatA(_In_ LPCSTR lpszFormat)                                                                                                                                        |
| wui         | UINT       | RegisterClipboardFormatW(_In_ LPCWSTR lpszFormat)                                                                                                                                       |
| ttuit       | HDEVNOTIFY | RegisterDeviceNotification(_In_ HANDLE hRecipient, _In_ LPVOID NotificationFilter, _In_ DWORD Flags)                                                                                    |
| ttuit       | HDEVNOTIFY | RegisterDeviceNotificationA(_In_ HANDLE hRecipient, _In_ LPVOID NotificationFilter, _In_ DWORD Flags)                                                                                   |
|             |            | RegisterDeviceNotificationW(_In_ HANDLE hRecipient, _In_                                                                                                                                |

|           |              |                                                                                                                                                                               |
|-----------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuit     | HDEVNOTIFY   | LPVOID NotificationFilter, _In_ DWORD Flags)                                                                                                                                  |
| tiuiiii   | BOOL         | RegisterHotKey(_In_opt_ HWND hWnd, _In_ int id, _In_ UINT fsModifiers, _In_ UINT vk)                                                                                          |
| ttuit     | HPOWERNOTIFY | RegisterPowerSettingNotification(_In_ HANDLE hRecipient, _In_ LPCGUID PowerSettingGuid, _In_ DWORD Flags)                                                                     |
| tuiiii    | BOOL         | RegisterRawInputDevices(_In_ PCRAWINPUTDEVICE pRawInputDevices, _In_ UINT uiNumDevices, _In_ UINT cbSize)                                                                     |
| ti        | BOOL         | RegisterShellHookWindow(_In_ HWND hwnd)                                                                                                                                       |
| tuii      | BOOL         | RegisterTouchWindow(_In_ HWND hWnd, _In_ ULONG ulFlags)                                                                                                                       |
| sui       | UINT         | RegisterWindowMessage(_In_ LPCTSTR lpString)                                                                                                                                  |
| au        | UINT         | RegisterWindowMessageA(_In_ LPCSTR lpString)                                                                                                                                  |
| wui       | UINT         | RegisterWindowMessageW(_In_ LPCWSTR lpString)                                                                                                                                 |
| i         | BOOL         | ReleaseCapture(void)                                                                                                                                                          |
| t         | int          | ReleaseDC(_In_ HWND hWnd, _In_ HDC hDC)                                                                                                                                       |
| ti        | BOOL         | RemoveClipboardFormatListener(_In_ HWND hwnd)                                                                                                                                 |
| tuiiii    | BOOL         | RemoveMenu(_In_ HMENU hMenu, _In_ UINT uPosition, _In_ UINT uFlags)                                                                                                           |
| tst       | HANDLE       | RemoveProg(_In_ HWND hWnd, _In_ LPCTSTR lpString)                                                                                                                             |
| t         | HANDLE       | RemoveProgA(_In_ HWND hWnd, _In_ LPCSTR lpString)                                                                                                                             |
| t         | HANDLE       | RemoveProgW(_In_ HWND hWnd, _In_ LPCWSTR lpString)                                                                                                                            |
| ti        | BOOL         | ReplyMessage(_In_ LRESULT lResult)                                                                                                                                            |
| tuiitt    | LPARAM       | ReuseDDElParam(_In_ LPARAM lParam, _In_ UINT msgIn, _In_ UINT msgOut, _In_ UINT_PTR uiLo, _In_ UINT_PTR uiHi)                                                                 |
| ti        | BOOL         | ScreenToClient(_In_ HWND hWnd, LPPOINT lpPoint)                                                                                                                               |
| tiittti   | BOOL         | ScrollDC(_In_ HDC hDC, _In_ int dx, _In_ int dy, _In_ const RECT *lprcScroll, _In_ const RECT *lprcClip, _In_ HRGN hrgnUpdate, _Out_ LPRECT lprcUpdate)                       |
| tiitti    | BOOL         | ScrollWindow(_In_ HWND hWnd, _In_ int XAmount, _In_ int YAmount, _In_ const RECT *lpRect, _In_ const RECT *lpClipRect)                                                        |
| tiitttiii | int          | ScrollWindowEx(_In_ HWND hWnd, _In_ int dx, _In_ int dy, _In_ const RECT *prcScroll, _In_ const RECT *prcClip, _In_ HRGN hrgnUpdate, _Out_ LPRECT prcUpdate, _In_ UINT flags) |
| tiuitt    | LRESULT      | SendDlgItemMessage(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                |
|           |              |                                                                                                                                                                               |

|            |         |                                                                                                                                                                    |
|------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuittt     | LRESULT | SendDlgItemMessageA(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                    |
| tuittt     | LRESULT | SendDlgItemMessageW(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                    |
| uitui      | UINT    | SendInput(_In_ UINT nInputs, _In_ LPINPUT pInputs, _In_ int cbSize)                                                                                                |
| tuittt     | LRESULT | SendMessage(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                 |
| tuittt     | LRESULT | SendMessageA(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                |
| tuitttti   | BOOL    | SendMessageCallback(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam, _In_ SENDASYNCPROC lpCallback, _In_ ULONG_PTR dwData)                   |
| tuitttti   | BOOL    | SendMessageCallbackA(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam, _In_ SENDASYNCPROC lpCallback, _In_ ULONG_PTR dwData)                  |
| tuitttti   | BOOL    | SendMessageCallbackW(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam, _In_ SENDASYNCPROC lpCallback, _In_ ULONG_PTR dwData)                  |
| tuittuiitt | LRESULT | SendMessageTimeout(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam, _In_ UINT fuFlags, _In_ UINT uTimeout, _Out_opt_ PDWORD_PTR lpdwResult)  |
| tuittuiitt | LRESULT | SendMessageTimeoutA(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam, _In_ UINT fuFlags, _In_ UINT uTimeout, _Out_opt_ PDWORD_PTR lpdwResult) |
| tuittuiitt | LRESULT | SendMessageTimeoutW(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam, _In_ UINT fuFlags, _In_ UINT uTimeout, _Out_opt_ PDWORD_PTR lpdwResult) |
| tuittt     | LRESULT | SendMessageW(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                                |
| tuitti     | BOOL    | SendNotifyMessage(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                           |
| tuitti     | BOOL    | SendNotifyMessageA(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                          |
| tuitti     | BOOL    | SendNotifyMessageW(_In_ HWND hWnd, _In_ UINT Msg, _In_ WPARAM wParam, _In_ LPARAM lParam)                                                                          |
| tt         | HWND    | SetActiveWindow(_In_ HWND hwnd)                                                                                                                                    |
| tt         | HWND    | SetCapture(_In_ HWND hwnd)                                                                                                                                         |

|          |         |                                                                                                                               |
|----------|---------|-------------------------------------------------------------------------------------------------------------------------------|
| uii      | BOOL    | SetCaretBlinkTime(_In_ UINT uMSeconds)                                                                                        |
| iii      | BOOL    | SetCaretPos(_In_ int X, _In_ int Y)                                                                                           |
| tiuiui   | DWORD   | SetClassLong(_In_ HWND hWnd, _In_ int nIndex, _In_ LONG dwNewLong)                                                            |
| tiuiui   | DWORD   | SetClassLongA(_In_ HWND hWnd, _In_ int nIndex, _In_ LONG dwNewLong)                                                           |
| tiuiui   | DWORD   | SetClassLongW(_In_ HWND hWnd, _In_ int nIndex, _In_ LONG dwNewLong)                                                           |
| tiuhuh   | WORD    | SetClassWord(_In_ HWND hWnd, _In_ int nIndex, _In_ WORD wNewWord)                                                             |
| uitt     | HANDLE  | SetClipboardData(_In_ UINT uFormat, _In_opt_ HANDLE hMem)                                                                     |
| tt       | HWND    | SetClipboardViewer(_In_ HWND hWndNewViewer)                                                                                   |
| tt       | HCURSOR | SetCursor(_In_opt_ HCURSOR hCursor)                                                                                           |
| iii      | BOOL    | SetCursorPos(_In_ int X, _In_ int Y)                                                                                          |
| tiuiii   | BOOL    | SetDlgItemInt(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ UINT uValue, _In_ BOOL bSigned)                                       |
| tisi     | BOOL    | SetDlgItemText(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ LPCTSTR lpString)                                                    |
| tiai     | BOOL    | SetDlgItemTextA(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ LPCSTR lpString)                                                    |
| tiwi     | BOOL    | SetDlgItemTextW(_In_ HWND hDlg, _In_ int nIDDlgItem, _In_ LPCWSTR lpString)                                                   |
| uii      | BOOL    | SetDoubleClickTime(_In_ UINT uInterval)                                                                                       |
| tt       | HWND    | SetFocus(_In_opt_ HWND hWnd)                                                                                                  |
| ti       | BOOL    | SetForegroundWindow(_In_ HWND hWnd)                                                                                           |
| tuiuitui | BOOL    | SetGestureConfig(_In_ HWND hWnd, _In_ DWORD dwReserved, _In_ UINT cIDs, _In_ PGESTURECONFIG pGestureConfig, _In_ UINT cbSize) |
| ti       | BOOL    | SetKeyboardState(_In_ LPBYTE lpKeyState)                                                                                      |
| uiuii    | VOID    | SetLastErrorEx(_In_ DWORD dwErrCode, _In_ DWORD dwType)                                                                       |
| tuiucui  | BOOL    | SetLayeredWindowAttributes(_In_ HWND hWnd, _In_ COLORREF crKey, _In_ BYTE bAlpha, _In_ DWORD dwFlags)                         |
| tii      | BOOL    | SetMenu(_In_ HWND hWnd, _In_opt_ HMENU hMenu)                                                                                 |
| tuii     | BOOL    | SetMenuContextHelpId(HMENU hMenu, DWORD dwContextHelpId)                                                                      |
| tuiiii   | BOOL    | SetMenuDefaultItem(_In_ HMENU hMenu, _In_ UINT uItem, _In_ UINT fByPos)                                                       |

|          |          |                                                                                                                                                 |
|----------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| tii      | BOOL     | SetMenuItem(_In_ HMENU hmenu, _In_ LPCMENUINFO lpcmi)                                                                                           |
| tuiiitti | BOOL     | SetMenuItemBitmaps(_In_ HMENU hMenu, _In_ UINT uPosition, _In_ UINT uFlags, _In_opt_ HBITMAP hBitmapUnchecked, _In_opt_ HBITMAP hBitmapChecked) |
| tuiiti   | BOOL     | SetMenuItemInfo(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _In_ LPMENUITEMINFO lpmii)                                            |
| tuiiti   | BOOL     | SetMenuItemInfoA(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _In_ LPMENUITEMINFO lpmii)                                           |
| tuiiti   | BOOL     | SetMenuItemInfoW(_In_ HMENU hMenu, _In_ UINT uItem, _In_ BOOL fByPosition, _In_ LPMENUITEMINFO lpmii)                                           |
| tt       | LPARAM   | SetMessageExtraInfo(_In_ LPARAM lParam)                                                                                                         |
| ttt      | HWND     | SetParent(_In_ HWND hWndChild, _In_opt_ HWND hWndNewParent)                                                                                     |
| iii      | BOOL     | SetPhysicalCursorPos(_In_ int X, _In_ int Y)                                                                                                    |
| uii      | BOOL     | SetProcessDefaultLayout(_In_ DWORD dwDefaultLayout)                                                                                             |
| i        | BOOL     | SetProcessDPIAware(void)                                                                                                                        |
| ti       | BOOL     | SetProcessWindowStation(_In_ HWINSTA hWinSta)                                                                                                   |
| tsti     | BOOL     | SetProp(_In_ HWND hWnd, _In_ LPCTSTR lpString, _In_opt_ HANDLE hData)                                                                           |
| tati     | BOOL     | SetPropA(_In_ HWND hWnd, _In_ LPCSTR lpString, _In_opt_ HANDLE hData)                                                                           |
| twti     | BOOL     | SetPropW(_In_ HWND hWnd, _In_ LPCWSTR lpString, _In_opt_ HANDLE hData)                                                                          |
| tiiii    | BOOL     | SetRect(_Out_ LPRECT lprc, _In_ int xLeft, _In_ int yTop, _In_ int xRight, _In_ int yBottom)                                                    |
| ti       | BOOL     | SetRectEmpty(_Out_ LPRECT lprc)                                                                                                                 |
| titi     | int      | SetScrollInfo(_In_ HWND hwnd, _In_ int fnBar, _In_ LPCSCROLLINFO lpsi, _In_ BOOL fRedraw)                                                       |
| tiii     | int      | SetScrollPos(_In_ HWND hWnd, _In_ int nBar, _In_ int nPos, _In_ BOOL bRedraw)                                                                   |
| tiiii    | BOOL     | SetScrollRange(_In_ HWND hWnd, _In_ int nBar, _In_ int nMinPos, _In_ int nMaxPos, _In_ BOOL bRedraw)                                            |
| itti     | BOOL     | SetSysColors(_In_ int cElements, _In_ const INT *lpaElements, _In_ const COLORREF *lpaRgbValues)                                                |
| tuii     | BOOL     | SetSystemCursor(_In_ HCURSOR hcur, _In_ DWORD id)                                                                                               |
| ti       | BOOL     | SetThreadDesktop(_In_ HDESK hDesktop)                                                                                                           |
| ttuitt   | UINT_PTR | SetTimer(_In_opt_ HWND hWnd, _In_ UINT_PTR nIDEvent, _In_ UINT uElapse, _In_opt_ TIMERPROC lpTimerFunc)                                         |

|               |               |                                                                                                                                                                                          |
|---------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tituii        | BOOL          | SetUserObjectInformation(_In_ HANDLE hObj, _In_ int nIndex, _In_ PVOID pvInfo, _In_ DWORD nLength)                                                                                       |
| tituii        | BOOL          | SetUserObjectInformationA(_In_ HANDLE hObj, _In_ int nIndex, _In_ PVOID pvInfo, _In_ DWORD nLength)                                                                                      |
| tituii        | BOOL          | SetUserObjectInformationW(_In_ HANDLE hObj, _In_ int nIndex, _In_ PVOID pvInfo, _In_ DWORD nLength)                                                                                      |
| tti           | BOOL          | SetUserObjectSecurity(_In_ HANDLE hObj, _In_ PSECURITY_INFORMATION pSIRequested, _In_ PSECURITY_DESCRIPTOR pSID)                                                                         |
| tuii          | BOOL          | SetWindowContextHelpId(HWND hwnd, DWORD dwContextHelpId)                                                                                                                                 |
| tuii          | BOOL          | SetWindowDisplayAffinity(_In_ HWND hWnd, _In_ DWORD dwAffinity)                                                                                                                          |
| tiuiui        | LONG          | SetWindowLong(_In_ HWND hWnd, _In_ int nIndex, _In_ LONG dwNewLong)                                                                                                                      |
| tiuiui        | LONG          | SetWindowLongA(_In_ HWND hWnd, _In_ int nIndex, _In_ LONG dwNewLong)                                                                                                                     |
| tiuiui        | LONG          | SetWindowLongW(_In_ HWND hWnd, _In_ int nIndex, _In_ LONG dwNewLong)                                                                                                                     |
| tti           | BOOL          | SetWindowPlacement(_In_ HWND hWnd, _In_ const WINDOWPLACEMENT *lpwndpl)                                                                                                                  |
| ttiiiuii      | BOOL          | SetWindowPos(_In_ HWND hWnd, _In_opt_ HWND hWndInsertAfter, _In_ int X, _In_ int Y, _In_ int cx, _In_ int cy, _In_ UINT uFlags)                                                          |
| tii           | int           | SetWindowRgn(_In_ HWND hWnd, _In_ HRGN hRgn, _In_ BOOL bRedraw)                                                                                                                          |
| ittuit        | HHOOK         | SetWindowsHookEx(_In_ int idHook, _In_ HOOKPROC lpfn, _In_ HINSTANCE hMod, _In_ DWORD dwThreadId)                                                                                        |
| ittuit        | HHOOK         | SetWindowsHookExA(_In_ int idHook, _In_ HOOKPROC lpfn, _In_ HINSTANCE hMod, _In_ DWORD dwThreadId)                                                                                       |
| ittuit        | HHOOK         | SetWindowsHookExW(_In_ int idHook, _In_ HOOKPROC lpfn, _In_ HINSTANCE hMod, _In_ DWORD dwThreadId)                                                                                       |
| tsi           | BOOL          | SetWindowText(_In_ HWND hWnd, _In_opt_ LPCTSTR lpString)                                                                                                                                 |
| tai           | BOOL          | SetWindowTextA(_In_ HWND hWnd, _In_opt_ LPCSTR lpString)                                                                                                                                 |
| twi           | BOOL          | SetWindowTextW(_In_ HWND hWnd, _In_opt_ LPCWSTR lpString)                                                                                                                                |
| uiiuttuiuiuit | HWINEVENTHOOK | SetWinEventHook(_In_ UINT eventMin, _In_ UINT eventMax, _In_ HMODULE hmodWinEventProc, _In_ WINEVENTPROC lpfnWinEventProc, _In_ DWORD idProcess, _In_ DWORD idThread, _In_ UINT dwflags) |

|           |      |                                                                                                                                                                                 |
|-----------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti        | BOOL | ShowCaret(_In_opt_ HWND hWnd)                                                                                                                                                   |
| ii        | int  | ShowCursor(_In_ BOOL bShow)                                                                                                                                                     |
| tii       | BOOL | ShowOwnedPopups(_In_ HWND hWnd, _In_ BOOL fShow)                                                                                                                                |
| tiii      | BOOL | ShowScrollBar(_In_ HWND hWnd, _In_ int wBar, _In_ BOOL bShow)                                                                                                                   |
| tii       | BOOL | ShowWindow(_In_ HWND hWnd, _In_ int nCmdShow)                                                                                                                                   |
| tii       | BOOL | ShowWindowAsync(_In_ HWND hWnd, _In_ int nCmdShow)                                                                                                                              |
| twi       | BOOL | ShutdownBlockReasonCreate(_In_ HWND hWnd, _In_ LPCWSTR pwszReason)                                                                                                              |
| ti        | BOOL | ShutdownBlockReasonDestroy(_In_ HWND hWnd)                                                                                                                                      |
| twti      | BOOL | ShutdownBlockReasonQuery(_In_ HWND hWnd, _Out_opt_ LPWSTR pwszBuff, _Inout_ DWORD *pcchBuff)                                                                                    |
| i         | BOOL | SoundSentry(void)                                                                                                                                                               |
| ttti      | BOOL | SubtractRect(_Out_ LPRECT lprcDst, _In_ const RECT *lprcSrc1, _In_ const RECT *lprcSrc2)                                                                                        |
| ii        | BOOL | SwapMouseButton(_In_ BOOL fSwap)                                                                                                                                                |
| ti        | BOOL | SwitchDesktop(_In_ HDESK hDesktop)                                                                                                                                              |
| tii       | VOID | SwitchToThisWindow(_In_ HWND hWnd, _In_ BOOL fAltTab)                                                                                                                           |
| uiuitui   | BOOL | SystemParametersInfo(_In_ UINT uiAction, _In_ UINT uiParam, _Inout_ PVOID pvParam, _In_ UINT fWinIni)                                                                           |
| uiuitui   | BOOL | SystemParametersInfoA(_In_ UINT uiAction, _In_ UINT uiParam, _Inout_ PVOID pvParam, _In_ UINT fWinIni)                                                                          |
| uiuitui   | BOOL | SystemParametersInfoW(_In_ UINT uiAction, _In_ UINT uiParam, _Inout_ PVOID pvParam, _In_ UINT fWinIni)                                                                          |
| tisiitui  | LONG | TabbedTextOut(_In_ HDC hDC, _In_ int X, _In_ int Y, _In_ LPCTSTR lpString, _In_ int nCount, _In_ int nTabPositions, _In_ const LPINT lpnTabStopPositions, _In_ int nTabOrigin)  |
| tiaaitui  | LONG | TabbedTextOutA(_In_ HDC hDC, _In_ int X, _In_ int Y, _In_ LPCSTR lpString, _In_ int nCount, _In_ int nTabPositions, _In_ const LPINT lpnTabStopPositions, _In_ int nTabOrigin)  |
| tiiwiitui | LONG | TabbedTextOutW(_In_ HDC hDC, _In_ int X, _In_ int Y, _In_ LPCWSTR lpString, _In_ int nCount, _In_ int nTabPositions, _In_ const LPINT lpnTabStopPositions, _In_ int nTabOrigin) |
| tuituituh | WORD | TileWindows(_In_opt_ HWND hwndParent, _In_ UINT wHow, _In_opt_ const RECT *lpRect, _In_ UINT cKids, _In_opt_ const HWND *lpKids)                                                |
| uiuittiii | int  | ToAscii(_In_ UINT uVirtKey, _In_ UINT uScanCode, _In_opt_ const BYTE *lpKeyState, _Out_ LPWORD lpChar, _In_ UINT uFlags)                                                        |

|             |      |                                                                                                                                                                  |
|-------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uiuittuiti  | int  | ToAsciiEx(_In_ UINT uVirtKey, _In_ UINT uScanCode, _In_opt_ const BYTE *lpKeyState, _Out_ LPWORD lpChar, _In_ UINT uFlags, _In_opt_ HKL dwhkl)                   |
| uiuitwiuii  | int  | ToUnicode(_In_ UINT wVirtKey, _In_ UINT wScanCode, _In_opt_ const BYTE *lpKeyState, _Out_ LPWSTR pwszBuff, _In_ int cchBuff, _In_ UINT wFlags)                   |
| uiuitwiuiti | int  | ToUnicodeEx(_In_ UINT wVirtKey, _In_ UINT wScanCode, _In_ const BYTE *lpKeyState, _Out_ LPWSTR pwszBuff, _In_ int cchBuff, _In_ UINT wFlags, _In_opt_ HKL dwhkl) |
| ti          | BOOL | TrackMouseEvent(_Inout_ LPTRACKMOUSEEVENT lpEventTrack)                                                                                                          |
| tuiiiitti   | BOOL | TrackPopupMenu(_In_ HMENU hMenu, _In_ UINT uFlags, _In_ int x, _In_ int y, _In_ int nReserved, _In_ HWND hWnd, _In_opt_ const RECT *prcRect)                     |
| tuiiitti    | BOOL | TrackPopupMenuEx(_In_ HMENU hMenu, _In_ UINT fuFlags, _In_ int x, _In_ int y, _In_ HWND hWnd, _In_opt_ LPTMPARAMS lptpm)                                         |
| ttti        | int  | TranslateAccelerator(_In_ HWND hWnd, _In_ HACCEL hAccTable, _In_ LPMMSG lpMsg)                                                                                   |
| ttti        | int  | TranslateAcceleratorA(_In_ HWND hWnd, _In_ HACCEL hAccTable, _In_ LPMMSG lpMsg)                                                                                  |
| ttti        | int  | TranslateAcceleratorW(_In_ HWND hWnd, _In_ HACCEL hAccTable, _In_ LPMMSG lpMsg)                                                                                  |
| tti         | BOOL | TranslateMdiSysAccel(_In_ HWND hWndClient, _In_ LPMMSG lpMsg)                                                                                                    |
| ti          | BOOL | TranslateMessage(_In_ const MSG *lpmsg)                                                                                                                          |
| ti          | BOOL | UnhookWindowsHookEx(_In_ HHOOK hhk)                                                                                                                              |
| ti          | BOOL | UnhookWinEvent(_In_ HWINEVENTHOOK hWinEventHook)                                                                                                                 |
| ttti        | BOOL | UnionRect(_Out_ LPRECT lprcDst, _In_ const RECT *lprcSrc1, _In_ const RECT *lprcSrc2)                                                                            |
| ti          | BOOL | UnloadKeyboardLayout(_In_ HKL hKL)                                                                                                                               |
| uittti      | BOOL | UnpackDDElParam(_In_ UINT msg, _In_ LPARAM lParam, _Out_ PUINT_PTR puiLo, _Out_ PUINT_PTR puiHi)                                                                 |
| sti         | BOOL | UnregisterClass(_In_ LPCTSTR lpClassName, _In_opt_ HINSTANCE hInstance)                                                                                          |
| ati         | BOOL | UnregisterClassA(_In_ LPCSTR lpClassName, _In_opt_ HINSTANCE hInstance)                                                                                          |
| wti         | BOOL | UnregisterClassW(_In_ LPCWSTR lpClassName, _In_opt_ HINSTANCE hInstance)                                                                                         |
| ti          | BOOL | UnregisterDeviceNotification(_In_ HDEVNOTIFY Handle)                                                                                                             |
|             |      |                                                                                                                                                                  |

|             |       |                                                                                                                                                                                                                              |
|-------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii         | BOOL  | UnregisterHotKey(_In_opt_ HWND hWnd, _In_ int id)                                                                                                                                                                            |
| ti          | BOOL  | UnregisterPowerSettingNotification(_In_ HPOWERNOTIFY Handle)                                                                                                                                                                 |
| ti          | BOOL  | UnregisterTouchWindow(_In_ HWND hWnd)                                                                                                                                                                                        |
| tttttuituii | BOOL  | UpdateLayeredWindow(_In_ HWND hWnd, _In_opt_ HDC hdcDst, _In_opt_ POINT *pptDst, _In_opt_ SIZE *psize, _In_opt_ HDC hdcSrc, _In_opt_ POINT *pptSrc, _In_ COLORREF crKey, _In_opt_ BLENDFUNCTION *pblend, _In_ DWORD dwFlags) |
| tii         | BOOL  | UpdateLayeredWindowIndirect(_In_ HWND hWnd, _In_ const UPDATELAYEREDWINDOWINFO *pULWInfo)                                                                                                                                    |
| ti          | BOOL  | UpdateWindow(_In_ HWND hWnd)                                                                                                                                                                                                 |
| ttii        | BOOL  | UserHandleGrantAccess(_In_ HANDLE hUserHandle, _In_ HANDLE hJob, _In_ BOOL bGrant)                                                                                                                                           |
| tii         | BOOL  | ValidateRect(_In_ HWND hWnd, _In_ const RECT *lpRect)                                                                                                                                                                        |
| tii         | BOOL  | ValidateRgn(_In_ HWND hWnd, _In_ HRGN hRgn)                                                                                                                                                                                  |
| yh          | SHORT | VkKeyScan(_In_ TCHAR ch)                                                                                                                                                                                                     |
| yh          | SHORT | VkKeyScanA(_In_ TCHAR ch)                                                                                                                                                                                                    |
| yth         | SHORT | VkKeyScanEx(_In_ TCHAR ch, _In_ HKL dwhkl)                                                                                                                                                                                   |
| yth         | SHORT | VkKeyScanExA(_In_ TCHAR ch, _In_ HKL dwhkl)                                                                                                                                                                                  |
| yth         | SHORT | VkKeyScanExW(_In_ TCHAR ch, _In_ HKL dwhkl)                                                                                                                                                                                  |
| yh          | SHORT | VkKeyScanW(_In_ TCHAR ch)                                                                                                                                                                                                    |
| tuiui       | DWORD | WaitForInputIdle(_In_ HANDLE hProcess, _In_ DWORD dwMilliseconds)                                                                                                                                                            |
| i           | BOOL  | WaitMessage(void)                                                                                                                                                                                                            |
| tt          | HWND  | WindowFromDC(_In_ HDC hdc)                                                                                                                                                                                                   |
| i6t         | HWND  | WindowFromPhysicalPoint(_In_ POINT Point)                                                                                                                                                                                    |
| i6t         | HWND  | WindowFromPoint(_In_ POINT Point)                                                                                                                                                                                            |
| tsuiti      | BOOL  | WinHelp(HWND hWndMain, LPCTSTR lpszHelp, UINT uCommand, ULONG_PTR dwData)                                                                                                                                                    |
| tauiti      | BOOL  | WinHelpA(HWND hWndMain, LPCSTR lpszHelp, UINT uCommand, ULONG_PTR dwData)                                                                                                                                                    |
| twuiti      | BOOL  | WinHelpW(HWND hWndMain, LPCWSTR lpszHelp, UINT uCommand, ULONG_PTR dwData)                                                                                                                                                   |
| ssti        | int   | wsprintf(_Out_ LPTSTR lpOut, _In_ LPCTSTR lpFmt, _In_ ...)                                                                                                                                                                   |
| aati        | int   | wsprintfA(_Out_ LPSTR lpOut, _In_ LPCSTR lpFmt, _In_ ...)                                                                                                                                                                    |
| wwti        | int   | wsprintfW(_Out_ LPWSTR lpOut, _In_ LPCWSTR lpFmt,                                                                                                                                                                            |

|      |     |                                                                             |
|------|-----|-----------------------------------------------------------------------------|
|      |     | _In_ ...)                                                                   |
| ssti | int | wvsprintf(_Out_ LPCTSTR lpOutput, _In_ LPCTSTR lpFmt, _In_ va_list arglist) |
| aati | int | wvsprintfA(_Out_ LPSTR lpOutput, _In_ LPCSTR lpFmt, _In_ va_list arglist)   |
| wwti | int | wvsprintfW(_Out_ LPWSTR lpOutput, _In_ LPCWSTR lpFmt, _In_ va_list arglist) |

## Userenv.dll

|          |         |                                                                                                                                                         |
|----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii      | BOOL    | CreateEnvironmentBlock(_Out_ LPVOID *lpEnvironment, _In_opt_ HANDLE hToken, _In_ BOOL bInherit)                                                         |
| wwwuii   | HRESULT | CreateProfile(_In_ LPCWSTR pszUserSid, _In_ LPCWSTR pszUserName, _Out_ LPWSTR pszProfilePath, _In_ DWORD cchProfilePath)                                |
| sssi     | BOOL    | DeleteProfile(_In_ LPCTSTR lpSidString, _In_opt_ LPCTSTR lpProfilePath, _In_opt_ LPCTSTR lpComputerName)                                                |
| aaai     | BOOL    | DeleteProfileA(_In_ LPCSTR lpSidString, _In_opt_ LPCSTR lpProfilePath, _In_opt_ LPCSTR lpComputerName)                                                  |
| wwwi     | BOOL    | DeleteProfileW(_In_ LPCWSTR lpSidString, _In_opt_ LPCWSTR lpProfilePath, _In_opt_ LPCWSTR lpComputerName)                                               |
| ti       | BOOL    | DestroyEnvironmentBlock(_In_ LPVOID lpEnvironment)                                                                                                      |
| it       | HANDLE  | EnterCriticalPolicySection(_In_ BOOL bMachine)                                                                                                          |
| tssuii   | BOOL    | ExpandEnvironmentStringsForUser(_In_opt_ HANDLE hToken, _In_ LPCTSTR lpSrc, _Out_ LPTSTR lpDest, _In_ DWORD dwSize)                                     |
| taauui   | BOOL    | ExpandEnvironmentStringsForUserA(_In_opt_ HANDLE hToken, _In_ LPCSTR lpSrc, _Out_ LPSTR lpDest, _In_ DWORD dwSize)                                      |
| twwuii   | BOOL    | ExpandEnvironmentStringsForUserW(_In_opt_ HANDLE hToken, _In_ LPCWSTR lpSrc, _Out_ LPWSTR lpDest, _In_ DWORD dwSize)                                    |
| ti       | BOOL    | FreeGPOList(_In_ PGROUP_POLICY_OBJECT pGPOList)                                                                                                         |
| ti       | BOOL    | FreeGPOListA(_In_ PGROUP_POLICY_OBJECT pGPOList)                                                                                                        |
| ti       | BOOL    | FreeGPOListW(_In_ PGROUP_POLICY_OBJECT pGPOList)                                                                                                        |
| sti      | BOOL    | GetAllUsersProfileDirectory(_Out_opt_ LPTSTR lpProfileDir, _Inout_ LPDWORD lpCchSize)                                                                   |
| ati      | BOOL    | GetAllUsersProfileDirectoryA(_Out_opt_ LPSTR lpProfileDir, _Inout_ LPDWORD lpCchSize)                                                                   |
| wti      | BOOL    | GetAllUsersProfileDirectoryW(_Out_opt_ LPWSTR lpProfileDir, _Inout_ LPDWORD lpCchSize)                                                                  |
| uistttui | DWORD   | GetAppliedGPOList(_In_ DWORD dwFlags, _In_ LPCTSTR pMachineName, _In_ PSID pSidUser, _In_ GUID *pGuidExtension, _Out_ PGROUP_POLICY_OBJECT *ppGPOList)  |
| uiatttui | DWORD   | GetAppliedGPOListA(_In_ DWORD dwFlags, _In_ LPCSTR pMachineName, _In_ PSID pSidUser, _In_ GUID *pGuidExtension, _Out_ PGROUP_POLICY_OBJECT *ppGPOList)  |
| uiwtttui | DWORD   | GetAppliedGPOListW(_In_ DWORD dwFlags, _In_ LPCWSTR pMachineName, _In_ PSID pSidUser, _In_ GUID *pGuidExtension, _Out_ PGROUP_POLICY_OBJECT *ppGPOList) |
| sti      | BOOL    | GetDefaultUserProfileDirectory(_Out_opt_ LPTSTR lpProfileDir, _Inout_                                                                                   |

|          |       |                                                                                                                                                                      |
|----------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          |       | LPDWORD lpchSize)                                                                                                                                                    |
| ati      | BOOL  | GetDefaultUserProfileDirectoryA(_Out_opt_ LPSTR lpProfileDir, _Inout_ LPDWORD lpchSize)                                                                              |
| wti      | BOOL  | GetDefaultUserProfileDirectoryW(_Out_opt_ LPWSTR lpProfileDir, _Inout_ LPDWORD lpchSize)                                                                             |
| tsssuiti | BOOL  | GetGPOList(_In_ HANDLE hToken, _In_ LPCTSTR lpName, _In_ LPCTSTR lpHostName, _In_ LPCTSTR lpComputerName, _In_ DWORD dwFlags, _Out_ PGROUP_POLICY_OBJECT *pGPOList)  |
| taaauiti | BOOL  | GetGPOListA(_In_ HANDLE hToken, _In_ LPCSTR lpName, _In_ LPCSTR lpHostName, _In_ LPCSTR lpComputerName, _In_ DWORD dwFlags, _Out_ PGROUP_POLICY_OBJECT *pGPOList)    |
| twwwuiti | BOOL  | GetGPOListW(_In_ HANDLE hToken, _In_ LPCWSTR lpName, _In_ LPCWSTR lpHostName, _In_ LPCWSTR lpComputerName, _In_ DWORD dwFlags, _Out_ PGROUP_POLICY_OBJECT *pGPOList) |
| sti      | BOOL  | GetProfilesDirectory(_Out_ LPTSTR lpProfilesDir, _Inout_ LPDWORD lpchSize)                                                                                           |
| ati      | BOOL  | GetProfilesDirectoryA(_Out_ LPSTR lpProfilesDir, _Inout_ LPDWORD lpchSize)                                                                                           |
| wti      | BOOL  | GetProfilesDirectoryW(_Out_ LPWSTR lpProfilesDir, _Inout_ LPDWORD lpchSize)                                                                                          |
| ti       | BOOL  | GetProfileType(_Out_ DWORD *pdwFlags)                                                                                                                                |
| tsti     | BOOL  | GetUserProfileDirectory(_In_ HANDLE hToken, _Out_opt_ LPTSTR lpProfileDir, _Inout_ LPDWORD lpchSize)                                                                 |
| tati     | BOOL  | GetUserProfileDirectoryA(_In_ HANDLE hToken, _Out_opt_ LPSTR lpProfileDir, _Inout_ LPDWORD lpchSize)                                                                 |
| twti     | BOOL  | GetUserProfileDirectoryW(_In_ HANDLE hToken, _Out_opt_ LPWSTR lpProfileDir, _Inout_ LPDWORD lpchSize)                                                                |
| ti       | BOOL  | LeaveCriticalPolicySection(_In_ HANDLE hSection)                                                                                                                     |
| tti      | BOOL  | LoadUserProfile(_In_ HANDLE hToken, _Inout_ LPPROFILEINFO lpProfileInfo)                                                                                             |
| tti      | BOOL  | LoadUserProfileA(_In_ HANDLE hToken, _Inout_ LPPROFILEINFO lpProfileInfo)                                                                                            |
| tti      | BOOL  | LoadUserProfileW(_In_ HANDLE hToken, _Inout_ LPPROFILEINFO lpProfileInfo)                                                                                            |
| ttuiui   | DWORD | ProcessGroupPolicyCompleted(_In_ REFGPEXTENSIONID extensionId, _In_ ASYNCCOMPLETIONHANDLE pAsyncHandle, _In_ DWORD dwStatus)                                         |
| ttuiui   | DWORD | ProcessGroupPolicyCompletedEx(_In_ REFGPEXTENSIONID extensionId, _In_ ASYNCCOMPLETIONHANDLE pAsyncHandle, _In_ DWORD dwStatus, _In_ HRESULT RsopStatus)              |
| ii       | BOOL  | RefreshPolicy(_In_ BOOL bMachine)                                                                                                                                    |
| iuii     | BOOL  | RefreshPolicyEx(_In_ BOOL bMachine, _In_ DWORD dwOptions)                                                                                                            |

|              |         |                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii          | BOOL    | RegisterGPNotification(_In_ HANDLE hEvent, _In_ BOOL bMachine)                                                                                                                                                                                                                                                                                                                                                        |
| tttuitttttti | HRESULT | RSOPAccessCheckByType(_In_ PSECURITY_DESCRIPTOR pSecurityDescriptor, _In_ PSID pPrincipalSelfSid, _In_ PRSOPTOKEN pRsopToken, _In_ DWORD dwDesiredAccessMask, _In_ POBJECT_TYPE_LIST pObjectTypeList, _In_ DWORD ObjectTypeListLength, _In_ PGENERIC_MAPPING pGenericMapping, _In_ PPRIVILEGE_SET pPrivilegeSet, _In_ LPDWORD pdwPrivilegeSetLength, _Out_ LPDWORD pdwGrantedAccessMask, _Out_ LPBOOL pbAccessStatus) |
| wtuitti      | HRESULT | RSOPFileAccessCheck(_In_ LPWSTR pszFileName, _In_ PRSOPTOKEN pRsopToken, _In_ DWORD dwDesiredAccessMask, _Out_ LPDWORD pdwGrantedAccessMask, _Out_ LPBOOL pbAccessStatus)                                                                                                                                                                                                                                             |
| uitti        | HRESULT | RSOPResetPolicySettingStatus(_In_ DWORD dwFlags, _In_ IWbemServices *pServices, _In_ IWbemClassObject *pSettingInstance)                                                                                                                                                                                                                                                                                              |
| uittuiti     | HRESULT | RSOPSetPolicySettingStatus(_In_ DWORD dwFlags, _In_ IWbemServices *pServices, _In_ IWbemClassObject *pSettingInstance, _In_ DWORD nInfo, _In_ POLICYSETTINGSTATUSINFO *pStatus)                                                                                                                                                                                                                                       |
| tii          | BOOL    | UnloadUserProfile(_In_ HANDLE hToken, _In_ HANDLE hProfile)                                                                                                                                                                                                                                                                                                                                                           |
| ti           | BOOL    | UnregisterGPNotification(_In_ HANDLE hEvent)                                                                                                                                                                                                                                                                                                                                                                          |

# UxTheme.dll

|              |                  |                                                                                                                                                                                                                         |
|--------------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuitttt     | HANIMATIONBUFFER | BeginBufferedAnimation (HWND hwnd, HDC hdcTarget, const RECT *rcTarget, BP_BUFFERFORMAT dwFormat, _In_ BP_PAINTPARAMS *pPaintParams, _In_ BP_ANIMATIONPARAMS *pAnimationParams, _Out_ HDC *phdcFrom, _Out_ HDC *phdcTo) |
| ttuittt      | HPAINTBUFFER     | BeginBufferedPaint (HDC hdcTarget, const RECT *prcTarget, BP_BUFFERFORMAT dwFormat, _In_ BP_PAINTPARAMS *pPaintParams, _Out_ HDC *phdc)                                                                                 |
| ti           | BOOL             | BeginPanningFeedback (_In_ HWND hwnd)                                                                                                                                                                                   |
| tii          | HRESULT          | BufferedPaintClear (HPAINTBUFFER hBufferedPaint, _In_ const RECT *prc)                                                                                                                                                  |
| i            | HRESULT          | BufferedPaintInit (void)                                                                                                                                                                                                |
| tii          | BOOL             | BufferedPaintRenderAnimation (HWND hwnd, HDC hdcTarget)                                                                                                                                                                 |
| ttuci        | HRESULT          | BufferedPaintSetAlpha (HPAINTBUFFER hBufferedPaint, _In_ const RECT *prc, BYTE alpha)                                                                                                                                   |
| ti           | HRESULT          | BufferedPaintStopAllAnimations (HWND hwnd)                                                                                                                                                                              |
| i            | HRESULT          | BufferedPaintUnInit (void)                                                                                                                                                                                              |
| ti           | HRESULT          | CloseThemeData (_In_ HTHEME hTheme)                                                                                                                                                                                     |
| ttiiiti      | HRESULT          | DrawThemeBackground (_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ const RECT *pRect, _In_ const RECT *pClipRect)                                                                         |
| ttiiiti      | HRESULT          | DrawThemeBackgroundEx (_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ const RECT *pRect, _In_ const DTBGOPTS *pOptions)                                                                    |
| ttituiuiti   | HRESULT          | DrawThemeEdge (_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCRECT pDestRect, _In_ UINT uEdge, _In_ UINT uFlags, _Out_ LPRECT pContentRect)                                             |
| ttiiitii     | HRESULT          | DrawThemeIcon (_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCRECT pRect, _In_ HIMAGELIST himl, _In_ int iImageIndex)                                                                   |
| ttii         | HRESULT          | DrawThemeParentBackground (_In_ HWND hwnd, _In_ HDC hdc, _In_ const RECT *prc)                                                                                                                                          |
| ttuiti       | HRESULT          | DrawThemeParentBackgroundEx (_In_ HWND hwnd, _In_ HDC hdc, _In_ DWORD dwFlags, _In_ const RECT *prc)                                                                                                                    |
| ttiiwiuiuiti | HRESULT          | DrawThemeText (_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCWSTR pszText, _In_ int iCharCount, _In_ DWORD dwTextFlags, _In_                                                           |

|            |         |                                                                                                                                                                                                           |
|------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |         | DWORD dwTextFlags2, _In_ LPCRECT pRect)                                                                                                                                                                   |
| tiiwiuitti | HRESULT | DrawThemeTextEx(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCWSTR pszText, _In_ int iCharCount, _In_ DWORD dwFlags, _Inout_ LPRECT pRect, _In_ const DTTOPTS *pOptions) |
| tuii       | HRESULT | EnableThemeDialogTexture(_In_ HWND hwnd, _In_ DWORD dwFlags)                                                                                                                                              |
| ii         | HRESULT | EnableTheming(_In_ BOOL fEnable)                                                                                                                                                                          |
| tii        | HRESULT | EndBufferedAnimation(HANIMATIONBUFFER hbpAnimation, BOOL fUpdateTarget)                                                                                                                                   |
| tii        | HRESULT | EndBufferedPaint(HPAINTBUFFER hBufferedPaint, BOOL fUpdateTarget)                                                                                                                                         |
| tii        | BOOL    | EndPanningFeedback(_In_ HWND hwnd, BOOL fAnimateBack)                                                                                                                                                     |
| titi       | HRESULT | GetBufferedPaintBit(HPAINTBUFFER hBufferedPaint, _Out_ RGBQUAD **ppbBuffer, _Out_ int *pcxRow)                                                                                                            |
| tt         | HDC     | GetBufferedPaintDC(HPAINTBUFFER hBufferedPaint)                                                                                                                                                           |
| tt         | HDC     | GetBufferedPaintTargetDC(HPAINTBUFFER hBufferedPaint)                                                                                                                                                     |
| titi       | HRESULT | GetBufferedPaintTargetRect(HPAINTBUFFER hBufferedPaint, _Out_ RECT *prc)                                                                                                                                  |
| wiwiwii    | HRESULT | GetCurrentThemeName(_Out_ LPWSTR pszThemeFileName, _In_ int dwMaxNameChars, _Out_ LPWSTR pszColorBuff, _In_ int cchMaxColorChars, _Out_ LPWSTR pszSizeBuff, _In_ int cchMaxSizeChars)                     |
| ui         | DWORD   | GetThemeAppProperties(void)                                                                                                                                                                               |
| titiiti    | HRESULT | GetThemeBackgroundContentRect(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCRECT pBoundingRect, _Out_ LPRECT pContentRect)                                               |
| titiiti    | HRESULT | GetThemeBackgroundExtent(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCRECT pContentRect, _Out_ LPRECT pExtentRect)                                                      |
| titiiti    | HRESULT | GetThemeBackgroundRegion(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCRECT pRect, _Out_ HRGN *pRegion)                                                                  |
| tiiiuiti   | HRESULT | GetThemeBitmap(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _In_ ULONG dwFlags, _Out_ HBITMAP *phBitmap)                                                                    |
| tiiiiti    | HRESULT | GetThemeBool(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ BOOL *pfVal)                                                                                                |
| tiiiiti    | HRESULT | GetThemeColor(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ COLORREF *pColor)                                                                                          |
|            |         | GetThemeDocumentationProperty(_In_ LPCWSTR                                                                                                                                                                |

|           |          |                                                                                                                                                                 |
|-----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wwwii     | HRESULT  | pszThemeName, _In_ LPCWSTR pszPropertyName, _Out_ LPWSTR pszValueBuff, _In_ int cchMaxValChars)                                                                 |
| tiiiti    | HRESULT  | GetThemeEnumValue(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ int *piVal)                                                  |
| tiiwii    | HRESULT  | GetThemeFilename(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ LPWSTR pszThemeFilename, _In_ int cchMaxBuffChars)            |
| tiiiiti   | HRESULT  | GetThemeFont(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ LOGFONTW *pFont)                                    |
| tiiiti    | HRESULT  | GetThemeInt(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ int *piVal)                                                        |
| tiiiti    | HRESULT  | GetThemeIntList(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ INTLIST *pIntList)                                             |
| tiiiiti   | HRESULT  | GetThemeMargins(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _In_ LPRECT prc, _Out_ MARGINS *pMargins)              |
| tiiiiti   | HRESULT  | GetThemeMetric(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ int *piVal)                                       |
| tiiituiti | HRESULT  | GetThemePartSize(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCRECT prc, _In_ THEMESIZE eSize, _Out_ SIZE *psz)                |
| tiiiti    | HRESULT  | GetThemePosition(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ POINT *pPoint)                                                |
| tiiiti    | HRESULT  | GetThemePropertyOrigin(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ PROPERTYORIGIN *pOrigin)                                |
| tiiiti    | HRESULT  | GetThemeRect(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ LPRECT pRect)                                                     |
| tiiittti  | HRESULT  | GetThemeStream(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ VOID **ppvStream, _Out_ DWORD *pcbStream, _In_ HINSTANCE hInst) |
| tiiwii    | HRESULT  | GetThemeString(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId, _In_ int iPropId, _Out_ LPWSTR pszBuff, _In_ int cchMaxBuffChars)                       |
| tii       | BOOL     | GetThemeSysBool(_In_ HTHEME hTheme, _In_ int iBoolID)                                                                                                           |
| tii       | COLORREF | GetThemeSysColor(_In_ HTHEME hTheme, _In_ int iColorID)                                                                                                         |
| tit       | HBRUSH   | GetThemeSysColorBrush(_In_ HTHEME hTheme, _In_ int iColorID)                                                                                                    |
| titi      | HRESULT  | GetThemeSysFont(_In_ HTHEME hTheme, _In_ int iFontID, _Out_ LOGFONTW *plf)                                                                                      |

|              |         |                                                                                                                                                                                                                    |
|--------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| titi         | HRESULT | GetThemeSysLot(_In_ HTHEME hTheme, _In_ int iIntID, _In_ int *piValue)                                                                                                                                             |
| tii          | int     | GetThemeSysSize(_In_ HTHEME hTheme, _In_ int iSizeID)                                                                                                                                                              |
| tiwii        | HRESULT | GetThemeSysString(_In_ HTHEME hTheme, _In_ int iStringID, _Out_ LPWSTR pszStringBuff, _In_ int cchMaxStringChars)                                                                                                  |
| ttiiwiuitti  | HRESULT | GetThemeTextExtent(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ LPCWSTR pszText, _In_ int iCharCount, _In_ DWORD dwTextFlags, _In_ LPCRECT pBoundingRect, _Out_ LPRECT pExtentRect) |
| ttiiti       | HRESULT | GetThemeTextMetrics(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _Out_ TEXTMETRIC *ptm)                                                                                                  |
| tiiiiti      | HRESULT | GetThemeTransitionDuration(HTHEME hTheme, int iPartId, int iStateIdFrom, int iStateIdTo, int iPropId, _Out_ DWORD *pdwDuration)                                                                                    |
| tt           | HTHEME  | GetWindowTheme(_In_ HWND hwnd)                                                                                                                                                                                     |
| ttiiuitti6ti | HRESULT | HitTestThemeBackground(_In_ HTHEME hTheme, _In_ HDC hdc, _In_ int iPartId, _In_ int iStateId, _In_ DWORD dwOptions, _In_ LPCRECT pRect, _In_ HRGN hrgn, _In_ POINT ptTest, _Out_ WORD *pwHitTestCode)              |
| i            | BOOL    | IsAppThemed(void)                                                                                                                                                                                                  |
| i            | BOOL    | IsCompositionActive(void)                                                                                                                                                                                          |
| i            | BOOL    | IsThemeActive(void)                                                                                                                                                                                                |
| tiii         | BOOL    | IsThemeBackgroundPartiallyTransparent(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId)                                                                                                                     |
| ti           | BOOL    | IsThemeDialogTextureEnabled(_In_ HWND hwnd)                                                                                                                                                                        |
| tiii         | BOOL    | IsThemePartDefined(_In_ HTHEME hTheme, _In_ int iPartId, _In_ int iStateId)                                                                                                                                        |
| twt          | HTHEME  | OpenThemeData(_In_ HWND hwnd, _In_ LPCWSTR pszClassList)                                                                                                                                                           |
| twuit        | HTHEME  | OpenThemeDataEx(_In_ HWND hwnd, _In_ LPCWSTR pszClassIdList, _In_ DWORD dwFlags)                                                                                                                                   |
| uii          | VOID    | SetThemeAppProperties(DWORD dwFlags)                                                                                                                                                                               |
| twwi         | HRESULT | SetWindowTheme(_In_ HWND hwnd, _In_ LPCWSTR pszSubAppName, _In_ LPCWSTR pszSubIdList)                                                                                                                              |
| tituii       | HRESULT | SetWindowThemeAttribute(_In_ HWND hwnd, _In_ enum WINDOWTHEMEATTRIBUTETYPE eAttribute, _In_ PVOID pvAttribute, _In_ DWORD cbAttribute)                                                                             |
| tuiuii       | BOOL    | UpdatePanningFeedback(_In_ HWND hwnd, _In_ LONG lTotalOverpanOffsetX, _In_ LONG lTotalOverpanOffsetY, _In_ BOOL fInertia)                                                                                          |

## Version.dll

|             |       |                                                                                                                                                                                                                       |
|-------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| suiiiti     | BOOL  | GetFileVersionInfo(_In_ LPCTSTR lptstrFilename, _Reserved_ DWORD dwHandle, _In_ DWORD dwLen, _Out_ LPVOID lpData)                                                                                                     |
| aiiuiiti    | BOOL  | GetFileVersionInfoA(_In_ LPCSTR lptstrFilename, _Reserved_ DWORD dwHandle, _In_ DWORD dwLen, _Out_ LPVOID lpData)                                                                                                     |
| uisuiiiti   | BOOL  | GetFileVersionInfoEx(_In_ DWORD dwFlags, _In_ LPCTSTR lptstrFilename, _Reserved_ DWORD dwHandle, _In_ DWORD dwLen, _Out_ LPVOID lpData)                                                                               |
| uiwuiiiti   | BOOL  | GetFileVersionInfoExW(_In_ DWORD dwFlags, _In_ LPCWSTR lptstrFilename, _Reserved_ DWORD dwHandle, _In_ DWORD dwLen, _Out_ LPVOID lpData)                                                                              |
| stui        | DWORD | GetFileVersionInfoSize(_In_ LPCTSTR lptstrFilename, _Out_opt_ LPDWORD lpdwHandle)                                                                                                                                     |
| atui        | DWORD | GetFileVersionInfoSizeA(_In_ LPCSTR lptstrFilename, _Out_opt_ LPDWORD lpdwHandle)                                                                                                                                     |
| uistui      | DWORD | GetFileVersionInfoSizeEx(_In_ DWORD dwFlags, _In_ LPCTSTR lptstrFilename, _Out_ LPDWORD lpdwHandle)                                                                                                                   |
| uiwtui      | DWORD | GetFileVersionInfoSizeExW(_In_ DWORD dwFlags, _In_ LPCWSTR lptstrFilename, _Out_ LPDWORD lpdwHandle)                                                                                                                  |
| wtui        | DWORD | GetFileVersionInfoSizeW(_In_ LPCWSTR lptstrFilename, _Out_opt_ LPDWORD lpdwHandle)                                                                                                                                    |
| wuiiiti     | BOOL  | GetFileVersionInfoW(_In_ LPCWSTR lptstrFilename, _Reserved_ DWORD dwHandle, _In_ DWORD dwLen, _Out_ LPVOID lpData)                                                                                                    |
| uissswtstui | DWORD | VerFindFile(_In_ DWORD dwFlags, _In_ LPCTSTR szFileName, _In_opt_ LPCTSTR szWinDir, _In_ LPCTSTR szAppDir, _Out_ LPWSTR szCurDir, _Inout_ PUINT lpuCurDirLen, _Out_ LPTSTR szDestDir, _Inout_ PUINT lpuDestDirLen)    |
| uiaaaatui   | DWORD | VerFindFileA(_In_ DWORD dwFlags, _In_ LPCSTR szFileName, _In_opt_ LPCSTR szWinDir, _In_ LPCSTR szAppDir, _Out_ LPWSTR szCurDir, _Inout_ PUINT lpuCurDirLen, _Out_ LPSTR szDestDir, _Inout_ PUINT lpuDestDirLen)       |
| uiwwwtwtui  | DWORD | VerFindFileW(_In_ DWORD dwFlags, _In_ LPCWSTR szFileName, _In_opt_ LPCWSTR szWinDir, _In_ LPCWSTR szAppDir, _Out_ LPWSTR szCurDir, _Inout_ PUINT lpuCurDirLen, _Out_ LPWSTR szDestDir, _Inout_ PUINT lpuDestDirLen)   |
| uissssstui  | DWORD | VerInstallFile(_In_ DWORD uFlags, _In_ LPCTSTR szSrcFileName, _In_ LPCTSTR szDestFileName, _In_ LPCTSTR szSrcDir, _In_ LPCTSTR szDestDir, _In_ LPCTSTR szCurDir, _Out_ LPTSTR szTmpFile, _Inout_ PUINT lpuTmpFileLen) |
| uiaaaaaatui | DWORD | VerInstallFileA(_In_ DWORD uFlags, _In_ LPCSTR szSrcFileName, _In_ LPCSTR szDestFileName, _In_ LPCSTR szSrcDir, _In_ LPCSTR                                                                                           |

|             |       |                                                                                                                                                                                                                        |
|-------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |       | szDestDir, _In_ LPCSTR szCurDir, _Out_ LPSTR szTmpFile, _Inout_ PUINT lpuTmpFileLen)                                                                                                                                   |
| uiwwwwwwtui | DWORD | VerInstallFileW(_In_ DWORD uFlags, _In_ LPCWSTR szSrcFileName, _In_ LPCWSTR szDestFileName, _In_ LPCWSTR szSrcDir, _In_ LPCWSTR szDestDir, _In_ LPCWSTR szCurDir, _Out_ LPWSTR szTmpFile, _Inout_ PUINT lpuTmpFileLen) |
| tssti       | BOOL  | VerQueryValue(_In_ LPCVOID pBlock, _In_ LPCTSTR lpSubBlock, _Out_ LPVOID *lplpBuffer, _Out_ PUINT puLen)                                                                                                               |
| tatti       | BOOL  | VerQueryValueA(_In_ LPCVOID pBlock, _In_ LPCSTR lpSubBlock, _Out_ LPVOID *lplpBuffer, _Out_ PUINT puLen)                                                                                                               |
| twtti       | BOOL  | VerQueryValueW(_In_ LPCVOID pBlock, _In_ LPCWSTR lpSubBlock, _Out_ LPVOID *lplpBuffer, _Out_ PUINT puLen)                                                                                                              |

# Winhttp.dll

|           |           |                                                                                                                                                                                                          |
|-----------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twuiiii   | BOOL      | WinHttpAddRequestHeaders(_In_ HINTERNET hRequest, _In_ LPCWSTR pwszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwModifiers)                                                                          |
| i         | BOOL      | WinHttpCheckPlatform(void)                                                                                                                                                                               |
| ti        | BOOL      | WinHttpCloseHandle(_In_ HINTERNET hInternet)                                                                                                                                                             |
| twuhuit   | HINTERNET | WinHttpConnect(_In_ HINTERNET hSession, _In_ LPCWSTR pwszServerName, _In_ INTERNET_PORT nServerPort, _Reserved_ DWORD dwReserved)                                                                        |
| wuiiiti   | BOOL      | WinHttpCrackUrl(_In_ LPCWSTR pwszUrl, _In_ DWORD dwUrlLength, _In_ DWORD dwFlags, _Inout_ LPURL_COMPONENTS lpUrlComponents)                                                                              |
| tuiwti    | BOOL      | WinHttpCreateUrl(_In_ LPURL_COMPONENTS lpUrlComponents, _In_ DWORD dwFlags, _Out_ LPWSTR pwszUrl, _Inout_ LPDWORD lpdwUrlLength)                                                                         |
| uiti      | BOOL      | WinHttpDetectAutoProxyConfigUrl(_In_ DWORD dwAutoDetectFlags, _Out_ LPWSTR *ppwszAutoConfigUrl)                                                                                                          |
| ti        | BOOL      | WinHttpGetDefaultProxyConfiguration(_Inout_ WINHTTP_PROXY_INFO *pProxyInfo)                                                                                                                              |
| ti        | BOOL      | WinHttpGetIEProxyConfigForCurrentUser(_Inout_ WINHTTP_CURRENT_USER_IE_PROXY_CONFIG *pProxyConfig)                                                                                                        |
| twtti     | BOOL      | WinHttpGetProxyForUrl(_In_ HINTERNET hSession, _In_ LPCWSTR lpcwszUrl, _In_ WINHTTP_AUTOPROXY_OPTIONS *pAutoProxyOptions, _Out_ WINHTTP_PROXY_INFO *pProxyInfo)                                          |
| wuiwwuit  | HINTERNET | WinHttpOpen(_In_opt_ LPCWSTR pwszUserAgent, _In_ DWORD dwAccessType, _In_ LPCWSTR pwszProxyName, _In_ LPCWSTR pwszProxyBypass, _In_ DWORD dwFlags)                                                       |
| twwwwtuit | HINTERNET | WinHttpOpenRequest(_In_ HINTERNET hConnect, _In_ LPCWSTR pwszVerb, _In_ LPCWSTR pwszObjectName, _In_ LPCWSTR pwszVersion, _In_ LPCWSTR pwszReferrer, _In_ LPCWSTR *ppwszAcceptTypes, _In_ DWORD dwFlags) |
| ttti      | BOOL      | WinHttpQueryAuthSchemes(_In_ HINTERNET hRequest, _Out_ LPDWORD lpdwSupportedSchemes, _Out_ LPDWORD lpdwFirstScheme, _Out_ LPDWORD pdwAuthTarget)                                                         |
| titi      | BOOL      | WinHttpQueryDataAvailable(_In_ HINTERNET hRequest, _Out_ LPDWORD lpdwNumberOfBytesAvailable)                                                                                                             |
| tuiwtiti  | BOOL      | WinHttpQueryHeaders(_In_ HINTERNET hRequest, _In_ DWORD dwInfoLevel, _In_opt_ LPCWSTR pwszName, _Out_ LPVOID lpBuffer, _Inout_ LPDWORD lpdwBufferLength,                                                 |

|             |                 |                                                                                                                                                                                                                    |
|-------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |                 | _Inout_ LPDWORD lpdwIndex)                                                                                                                                                                                         |
| tuitti      | BOOL            | WinHttpQueryOption(_In_ HINTERNET hInternet, _In_ DWORD dwOption, _Out_ LPVOID lpBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                                                         |
| ttuiti      | BOOL            | WinHttpReadData(_In_ HINTERNET hRequest, _Out_ LPVOID lpBuffer, _In_ DWORD dwNumberOfBytesToRead, _Out_ LPDWORD lpdwNumberOfBytesRead)                                                                             |
| titi        | BOOL            | WinHttpReceiveResponse(_In_ HINTERNET hRequest, _Reserved_ LPVOID lpReserved)                                                                                                                                      |
| twuituiuiti | BOOL            | WinHttpSendRequest(_In_ HINTERNET hRequest, _In_opt_ LPCWSTR pwszHeaders, _In_ DWORD dwHeadersLength, _In_opt_ LPVOID lpOptional, _In_ DWORD dwOptionalLength, _In_ DWORD dwTotalLength, _In_ DWORD_PTR dwContext) |
| tuiuiwwti   | BOOL            | WinHttpSetCredentials(_In_ HINTERNET hRequest, _In_ DWORD AuthTargets, _In_ DWORD AuthScheme, _In_ LPCWSTR pwszUserName, _In_ LPCWSTR pwszPassword, _Reserved_ LPVOID pAuthParams)                                 |
| ti          | BOOL            | WinHttpSetDefaultProxyConfiguration(_In_ WINHTTP_PROXY_INFO *pProxyInfo)                                                                                                                                           |
| tuituii     | BOOL            | WinHttpSetOption(_In_ HINTERNET hInternet, _In_ DWORD dwOption, _In_ LPVOID lpBuffer, _In_ DWORD dwBufferLength)                                                                                                   |
| ttuitt      | WINHTTP_STATUS_ | WinHttpSetStatusCallback(_In_ HINTERNET hInternet, _In_ WINHTTP_STATUS_CALLBACK lpfnInternetCallback, _In_ DWORD dwNotificationFlags, _Reserved_ DWORD_PTR dwReserved)                                             |
| tiiii       | BOOL            | WinHttpSetTimeouts(_In_ HINTERNET hInternet, _In_ int dwResolveTimeout, _In_ int dwConnectTimeout, _In_ int dwSendTimeout, _In_ int dwReceiveTimeout)                                                              |
| twi         | BOOL            | WinHttpTimeFromSystemTime(_In_ const SYSTEMTIME *pst, _Out_ LPWSTR pwszTime)                                                                                                                                       |
| wti         | BOOL            | WinHttpTimeToSystemTime(_In_ LPCWSTR pwszTime, _Out_ SYSTEMTIME *pst)                                                                                                                                              |
| ttuiti      | BOOL            | WinHttpWriteData(_In_ HINTERNET hRequest, _In_ LPCVOID lpBuffer, _In_ DWORD dwNumberOfBytesToWrite, _Out_ LPDWORD lpdwNumberOfBytesWritten)                                                                        |

# Wininet.dll

|              |         |                                                                                                                                                                                                                                                                                                               |
|--------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| aattuituiaai | BOOL    | <a href="#">CommitUrlCacheEntryA</a> (_In_ LPCSTR lpszUrlName, _In_ LPCSTR lpszLocalFileName, _In_ FILETIME ExpireTime, _In_ FILETIME LastModifiedTime, _In_ DWORD CacheEntryType, _In_ LPBYTE lpHeaderInfo, _In_ DWORD cchHeaderInfo, _Reserved_ LPCSTR lpszFileExtension, _In_ LPCSTR lpszOriginalUrl)      |
| wwttuiwuiwwi | BOOL    | <a href="#">CommitUrlCacheEntryW</a> (_In_ LPCWSTR lpszUrlName, _In_ LPCWSTR lpszLocalFileName, _In_ FILETIME ExpireTime, _In_ FILETIME LastModifiedTime, _In_ DWORD CacheEntryType, _In_ LPCWSTR lpHeaderInfo, _In_ DWORD cchHeaderInfo, _Reserved_ LPCWSTR lpszFileExtension, _In_ LPCWSTR lpszOriginalUrl) |
| tttti        | BOOL    | <a href="#">CreateMD5SSOHash</a> (_In_ PWSTR pszChallengeInfo, _In_ PWSTR pwszRealm, _In_ PWSTR pwszTarget, _Out_ PBYTE pbHexHash)                                                                                                                                                                            |
| suissuii     | BOOL    | <a href="#">CreateUrlCacheEntry</a> (_In_ LPCTSTR lpszUrlName, _In_ DWORD dwExpectedFileSize, _In_ LPCTSTR lpszFileExtension, _Out_ LPTSTR lpszFileName, _Reserved_ DWORD dwReserved)                                                                                                                         |
| auiaauui     | BOOL    | <a href="#">CreateUrlCacheEntryA</a> (_In_ LPCSTR lpszUrlName, _In_ DWORD dwExpectedFileSize, _In_ LPCSTR lpszFileExtension, _Out_ LPSTR lpszFileName, _Reserved_ DWORD dwReserved)                                                                                                                           |
| wuiwwuii     | BOOL    | <a href="#">CreateUrlCacheEntryW</a> (_In_ LPCWSTR lpszUrlName, _In_ DWORD dwExpectedFileSize, _In_ LPCWSTR lpszFileExtension, _Out_ LPWSTR lpszFileName, _Reserved_ DWORD dwReserved)                                                                                                                        |
| uiti6        | GROUPID | <a href="#">CreateUrlCacheGroup</a> (_In_ DWORD dwFlags, _Reserved_ LPVOID lpReserved)                                                                                                                                                                                                                        |
| si           | BOOL    | <a href="#">DeleteUrlCacheEntry</a> (_In_ LPCTSTR lpszUrlName)                                                                                                                                                                                                                                                |
| ai           | BOOL    | <a href="#">DeleteUrlCacheEntryA</a> (_In_ LPCSTR lpszUrlName)                                                                                                                                                                                                                                                |
| wi           | BOOL    | <a href="#">DeleteUrlCacheEntryW</a> (_In_ LPCWSTR lpszUrlName)                                                                                                                                                                                                                                               |
| i6uiti       | BOOL    | <a href="#">DeleteUrlCacheGroup</a> (_In_ GROUPID GroupId, _In_ DWORD dwFlags, _Reserved_ LPVOID lpReserved)                                                                                                                                                                                                  |
| auuii        | BOOL    | <a href="#">DetectAutoProxyUrl</a> (_Inout_ LPSTR lpszAutoProxyUrl, _In_ DWORD dwAutoProxyUrlLength, _In_ DWORD dwDetectFlags)                                                                                                                                                                                |
| ti           | BOOL    | <a href="#">FindCloseUrlCache</a> (_In_ HANDLE hEnumHandle)                                                                                                                                                                                                                                                   |
|              |         | <a href="#">FindFirstUrlCacheEntry</a> (_In_ LPCTSTR lpszUrlSearchPattern, _Out_                                                                                                                                                                                                                              |

|              |        |                                                                                                                                                                                                                                                                                                                                                     |
|--------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sttt         | HANDLE | LPINTERNET_CACHE_ENTRY_INFO<br>lpFirstCacheEntryInfo, _Inout_ LPDWORD<br>lpcbCacheEntryInfo)                                                                                                                                                                                                                                                        |
| attt         | HANDLE | FindFirstUrlCacheEntryA(_In_ LPCSTR<br>lpszUrlSearchPattern, _Out_<br>LPINTERNET_CACHE_ENTRY_INFO<br>lpFirstCacheEntryInfo, _Inout_ LPDWORD<br>lpcbCacheEntryInfo)                                                                                                                                                                                  |
| suiiii6ttttt | HANDLE | FindFirstUrlCacheEntryEx(_In_ LPCTSTR<br>lpszUrlSearchPattern, _In_ DWORD dwFlags, _In_<br>DWORD dwFilter, _In_ GROUPID GroupId, _Out_<br>LPINTERNET_CACHE_ENTRY_INFO<br>lpFirstCacheEntryInfo, _Inout_ LPDWORD lpdwEntryInfo,<br>_Reserved_ LPVOID lpGroupAttributes, _Reserved_<br>LPDWORD lpcbGroupAttributes, _Reserved_ LPVOID<br>lpReserved)  |
| aiuiii6ttttt | HANDLE | FindFirstUrlCacheEntryExA(_In_ LPCSTR<br>lpszUrlSearchPattern, _In_ DWORD dwFlags, _In_<br>DWORD dwFilter, _In_ GROUPID GroupId, _Out_<br>LPINTERNET_CACHE_ENTRY_INFO<br>lpFirstCacheEntryInfo, _Inout_ LPDWORD lpdwEntryInfo,<br>_Reserved_ LPVOID lpGroupAttributes, _Reserved_<br>LPDWORD lpcbGroupAttributes, _Reserved_ LPVOID<br>lpReserved)  |
| wuiiii6ttttt | HANDLE | FindFirstUrlCacheEntryExW(_In_ LPCWSTR<br>lpszUrlSearchPattern, _In_ DWORD dwFlags, _In_<br>DWORD dwFilter, _In_ GROUPID GroupId, _Out_<br>LPINTERNET_CACHE_ENTRY_INFO<br>lpFirstCacheEntryInfo, _Inout_ LPDWORD lpdwEntryInfo,<br>_Reserved_ LPVOID lpGroupAttributes, _Reserved_<br>LPDWORD lpcbGroupAttributes, _Reserved_ LPVOID<br>lpReserved) |
| wttt         | HANDLE | FindFirstUrlCacheEntryW(_In_ LPCWSTR<br>lpszUrlSearchPattern, _Out_<br>LPINTERNET_CACHE_ENTRY_INFO<br>lpFirstCacheEntryInfo, _Inout_ LPDWORD<br>lpcbCacheEntryInfo)                                                                                                                                                                                 |
| uiuituittt   | HANDLE | FindFirstUrlCacheGroup(_Reserved_ DWORD dwFlags,<br>_In_ DWORD dwFilter, _Reserved_ LPVOID<br>lpSearchCondition, _Reserved_ DWORD<br>dwSearchCondition, _Out_ GROUPID *lpGroupId,<br>_Reserved_ LPVOID lpReserved)                                                                                                                                  |
| ttti         | BOOL   | FindNextUrlCacheEntry(_In_ HANDLE hEnumHandle,<br>_Out_ LPINTERNET_CACHE_ENTRY_INFO<br>lpNextCacheEntryInfo, _Inout_ LPDWORD<br>lpcbCacheEntryInfo)                                                                                                                                                                                                 |
|              |        | FindNextUrlCacheEntryA(_In_ HANDLE hEnumHandle,<br>_Out_ LPINTERNET_CACHE_ENTRY_INFO                                                                                                                                                                                                                                                                |

|          |      |                                                                                                                                                                                                                                                       |
|----------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttti     | BOOL | lpNextCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo)                                                                                                                                                                                             |
| ttttti   | BOOL | FindNextUrlCacheEntryEx(_In_ HANDLE hEnumHandle, _Inout_ LPINTERNET_CACHE_ENTRY_INFO lpNextCacheEntryInfo, _Inout_ LPDWORD lpcbEntryInfo, _Reserved_ LPVOID lpGroupAttributes, _Reserved_ LPDWORD lpcbGroupAttributes, _Reserved_ LPVOID lpReserved)  |
| ttttti   | BOOL | FindNextUrlCacheEntryExA(_In_ HANDLE hEnumHandle, _Inout_ LPINTERNET_CACHE_ENTRY_INFO lpNextCacheEntryInfo, _Inout_ LPDWORD lpcbEntryInfo, _Reserved_ LPVOID lpGroupAttributes, _Reserved_ LPDWORD lpcbGroupAttributes, _Reserved_ LPVOID lpReserved) |
| ttttti   | BOOL | FindNextUrlCacheEntryExW(_In_ HANDLE hEnumHandle, _Inout_ LPINTERNET_CACHE_ENTRY_INFO lpNextCacheEntryInfo, _Inout_ LPDWORD lpcbEntryInfo, _Reserved_ LPVOID lpGroupAttributes, _Reserved_ LPDWORD lpcbGroupAttributes, _Reserved_ LPVOID lpReserved) |
| ttti     | BOOL | FindNextUrlCacheEntryW(_In_ HANDLE hEnumHandle, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpNextCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo)                                                                                                           |
| ttti     | BOOL | FindNextUrlCacheGroup(_In_ HANDLE hFind, _Out_ GROUPID *lpGroupId, _Reserved_ LPVOID lpReserved)                                                                                                                                                      |
| tiuiستي  | BOOL | FtpCommand(_In_ HINTERNET hConnect, _In_ BOOL fExpectResponse, _In_ DWORD dwFlags, _In_ LPCTSTR lpszCommand, _In_ DWORD_PTR dwContext, _Out_ HINTERNET *phFtpCommand)                                                                                 |
| tiuiاتي  | BOOL | FtpCommandA(_In_ HINTERNET hConnect, _In_ BOOL fExpectResponse, _In_ DWORD dwFlags, _In_ LPCSTR lpszCommand, _In_ DWORD_PTR dwContext, _Out_ HINTERNET *phFtpCommand)                                                                                 |
| tiuiwtتي | BOOL | FtpCommandW(_In_ HINTERNET hConnect, _In_ BOOL fExpectResponse, _In_ DWORD dwFlags, _In_ LPCWSTR lpszCommand, _In_ DWORD_PTR dwContext, _Out_ HINTERNET *phFtpCommand)                                                                                |
| tsi      | BOOL | FtpCreateDirectory(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszDirectory)                                                                                                                                                                               |
| tai      | BOOL | FtpCreateDirectoryA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszDirectory)                                                                                                                                                                               |
| twi      | BOOL | FtpCreateDirectoryW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszDirectory)                                                                                                                                                                              |

|            |           |                                                                                                                                                                                                                  |
|------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tsi        | BOOL      | <code>FtpDeleteFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszFileName)</code>                                                                                                                                   |
| tai        | BOOL      | <code>FtpDeleteFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszFileName)</code>                                                                                                                                   |
| twi        | BOOL      | <code>FtpDeleteFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszFileName)</code>                                                                                                                                  |
| tstuitt    | HINTERNET | <code>FtpFindFirstFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszSearchFile, _Out_ LPWIN32_FIND_DATA lpFindFileData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                        |
| tatuitt    | HINTERNET | <code>FtpFindFirstFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszSearchFile, _Out_ LPWIN32_FIND_DATA lpFindFileData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                        |
| twtuitt    | HINTERNET | <code>FtpFindFirstFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszSearchFile, _Out_ LPWIN32_FIND_DATA lpFindFileData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                       |
| tsti       | BOOL      | <code>FtpGetCurrentDirectory(_In_ HINTERNET hConnect, _Out_ LPTSTR lpszCurrentDirectory, _Inout_ LPDWORD lpdwCurrentDirectory)</code>                                                                            |
| tati       | BOOL      | <code>FtpGetCurrentDirectoryA(_In_ HINTERNET hConnect, _Out_ LPSTR lpszCurrentDirectory, _Inout_ LPDWORD lpdwCurrentDirectory)</code>                                                                            |
| twti       | BOOL      | <code>FtpGetCurrentDirectoryW(_In_ HINTERNET hConnect, _Out_ LPWSTR lpszCurrentDirectory, _Inout_ LPDWORD lpdwCurrentDirectory)</code>                                                                           |
| tssiuiuiti | BOOL      | <code>FtpGetFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszRemoteFile, _In_ LPCTSTR lpszNewFile, _In_ BOOL fFailIfExists, _In_ DWORD dwFlagsAndAttributes, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>  |
| taaiuiuiti | BOOL      | <code>FtpGetFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszRemoteFile, _In_ LPCSTR lpszNewFile, _In_ BOOL fFailIfExists, _In_ DWORD dwFlagsAndAttributes, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>   |
| ttui       | DWORD     | <code>FtpGetFileSize(_In_ HINTERNET hFile, _Out_ LPDWORD lpdwFileSizeHigh)</code>                                                                                                                                |
| twwiuiuiti | BOOL      | <code>FtpGetFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszRemoteFile, _In_ LPCWSTR lpszNewFile, _In_ BOOL fFailIfExists, _In_ DWORD dwFlagsAndAttributes, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code> |
| tsuiuitt   | HINTERNET | <code>FtpOpenFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszFileName, _In_ DWORD dwAccess, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                                                  |
|            |           |                                                                                                                                                                                                                  |

|         |           |                                                                                                                                                                |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tauiutt | HINTERNET | <code>FtpOpenFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszFileName, _In_ DWORD dwAccess, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                |
| twuiutt | HINTERNET | <code>FtpOpenFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszFileName, _In_ DWORD dwAccess, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>               |
| tssuiti | BOOL      | <code>FtpPutFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszLocalFile, _In_ LPCTSTR lpszNewRemoteFile, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>     |
| taauiti | BOOL      | <code>FtpPutFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszLocalFile, _In_ LPCSTR lpszNewRemoteFile, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>      |
| twwuiti | BOOL      | <code>FtpPutFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszLocalFile, _In_ LPCWSTR lpszNewRemoteFile, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>    |
| tsi     | BOOL      | <code>FtpRemoveDirectory(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszDirectory)</code>                                                                           |
| tai     | BOOL      | <code>FtpRemoveDirectoryA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszDirectory)</code>                                                                           |
| twi     | BOOL      | <code>FtpRemoveDirectoryW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszDirectory)</code>                                                                          |
| tssi    | BOOL      | <code>FtpRenameFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszExisting, _In_ LPCTSTR lpszNew)</code>                                                           |
| taai    | BOOL      | <code>FtpRenameFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszExisting, _In_ LPCSTR lpszNew)</code>                                                            |
| twwi    | BOOL      | <code>FtpRenameFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszExisting, _In_ LPCWSTR lpszNew)</code>                                                          |
| tsi     | BOOL      | <code>FtpSetCurrentDirectory(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszDirectory)</code>                                                                       |
| tai     | BOOL      | <code>FtpSetCurrentDirectoryA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszDirectory)</code>                                                                       |
| twi     | BOOL      | <code>FtpSetCurrentDirectoryW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszDirectory)</code>                                                                      |
| ttuii   | BOOL      | <code>GetUrlCacheConfigInfo(_Inout_ LPINTERNET_CACHE_CONFIG_INFO lpCacheConfigInfo, _Reserved_ LPDWORD lpcbCacheConfigInfo, _In_ DWORD dwFieldControl)</code>  |
| ttuii   | BOOL      | <code>GetUrlCacheConfigInfoA(_Inout_ LPINTERNET_CACHE_CONFIG_INFO lpCacheConfigInfo, _Reserved_ LPDWORD lpcbCacheConfigInfo, _In_ DWORD dwFieldControl)</code> |
| ttuii   | BOOL      | <code>GetUrlCacheConfigInfoW(_Inout_ LPINTERNET_CACHE_CONFIG_INFO lpCacheConfigInfo, _Reserved_ LPDWORD lpcbCacheConfigInfo, _In_ DWORD dwFieldControl)</code> |

|            |      |                                                                                                                                                                                                                                                                          |
|------------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |      | lpcbCacheConfigInfo, _In_ DWORD dwFieldControl)                                                                                                                                                                                                                          |
| stti       | BOOL | GetUrlCacheEntryInfo(_In_ LPCTSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo)                                                                                                                                   |
| atti       | BOOL | GetUrlCacheEntryInfoA(_In_ LPCSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo)                                                                                                                                   |
| sttsttuii  | BOOL | GetUrlCacheEntryInfoEx(_In_ LPCTSTR lpszUrl, _Inout_opt_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_opt_ LPDWORD lpcbCacheEntryInfo, _Reserved_ LPTSTR lpszRedirectUrl, _Reserved_ LPDWORD lpcbRedirectUrl, _Reserved_ LPVOID lpReserved, _In_ DWORD dwFlags)  |
| attattuii  | BOOL | GetUrlCacheEntryInfoExA(_In_ LPCSTR lpszUrl, _Inout_opt_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_opt_ LPDWORD lpcbCacheEntryInfo, _Reserved_ LPSTR lpszRedirectUrl, _Reserved_ LPDWORD lpcbRedirectUrl, _Reserved_ LPVOID lpReserved, _In_ DWORD dwFlags)   |
| wttwtuii   | BOOL | GetUrlCacheEntryInfoExW(_In_ LPCWSTR lpszUrl, _Inout_opt_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_opt_ LPDWORD lpcbCacheEntryInfo, _Reserved_ LPWSTR lpszRedirectUrl, _Reserved_ LPDWORD lpcbRedirectUrl, _Reserved_ LPVOID lpReserved, _In_ DWORD dwFlags) |
| wtti       | BOOL | GetUrlCacheEntryInfoW(_In_ LPCWSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo)                                                                                                                                  |
| i6uiiuitti | BOOL | GetUrlCacheGroupAttribute(_In_ GROUPID gid, _Reserved_ DWORD dwFlags, _In_ DWORD dwAttributes, _Out_ LPINTERNET_CACHE_GROUP_INFO lpGroupInfo, _Inout_ LPDWORD lpdwGroupInfo, _Reserved_ LPVOID lpReserved)                                                               |
| i6uiiuitti | BOOL | GetUrlCacheGroupAttributeA(_In_ GROUPID gid, _Reserved_ DWORD dwFlags, _In_ DWORD dwAttributes, _Out_ LPINTERNET_CACHE_GROUP_INFO lpGroupInfo, _Inout_ LPDWORD lpdwGroupInfo, _Reserved_ LPVOID lpReserved)                                                              |
| i6uiiuitti | BOOL | GetUrlCacheGroupAttributeW(_In_ GROUPID gid, _Reserved_ DWORD dwFlags, _In_ DWORD dwAttributes, _Out_ LPINTERNET_CACHE_GROUP_INFO lpGroupInfo, _Inout_ LPDWORD lpdwGroupInfo, _Reserved_ LPVOID lpReserved)                                                              |
|            |      | CacheCreateLocator(_In_ LPCTSTR lpszHost, _In_ INTERNET_PORT nServerPort, _In_ LPCTSTR                                                                                                                                                                                   |

|            |           |                                                                                                                                                                                                                                                                                        |
|------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| suhssuisti | BOOL      | <code>lpszDisplayString, _In_ LPCTSTR lpszSelectorString, _In_ DWORD dwGopherType, _Out_ LPCTSTR lpszLocator, _Inout_ LPDWORD lpdwBufferLength)</code>                                                                                                                                 |
| auhaauiati | BOOL      | <code>GopherCreateLocatorA(_In_ LPCSTR lpszHost, _In_ INTERNET_PORT nServerPort, _In_ LPCSTR lpszDisplayString, _In_ LPCSTR lpszSelectorString, _In_ DWORD dwGopherType, _Out_ LPCTSTR lpszLocator, _Inout_ LPDWORD lpdwBufferLength)</code>                                           |
| wuhwwuiwti | BOOL      | <code>GopherCreateLocatorW(_In_ LPCWSTR lpszHost, _In_ INTERNET_PORT nServerPort, _In_ LPCWSTR lpszDisplayString, _In_ LPCWSTR lpszSelectorString, _In_ DWORD dwGopherType, _Out_ LPWSTR lpszLocator, _Inout_ LPDWORD lpdwBufferLength)</code>                                         |
| tsstuiitt  | HINTERNET | <code>GopherFindFirstFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszLocator, _In_ LPCTSTR lpszSearchString, _Out_ LPGOPHER_FIND_DATA lpFindData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                                                                  |
| taatuitt   | HINTERNET | <code>GopherFindFirstFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszLocator, _In_ LPCSTR lpszSearchString, _Out_ LPGOPHER_FIND_DATA lpFindData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                                                                   |
| twwtuitt   | HINTERNET | <code>GopherFindFirstFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszLocator, _In_ LPCWSTR lpszSearchString, _Out_ LPGOPHER_FIND_DATA lpFindData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)</code>                                                                                 |
| tsstuititi | BOOL      | <code>GopherGetAttribute(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszLocator, _In_ LPCTSTR lpszAttributeName, _Out_ LPBYTE lpBuffer, _In_ DWORD dwBufferLength, _Out_ LPDWORD lpdwCharactersReturned, _In_ GOPHER_ATTRIBUTE_ENUMERATOR lpfmEnumerator, _In_ DWORD_PTR dwContext)</code>  |
| taatuititi | BOOL      | <code>GopherGetAttributeA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszLocator, _In_ LPCSTR lpszAttributeName, _Out_ LPBYTE lpBuffer, _In_ DWORD dwBufferLength, _Out_ LPDWORD lpdwCharactersReturned, _In_ GOPHER_ATTRIBUTE_ENUMERATOR lpfmEnumerator, _In_ DWORD_PTR dwContext)</code>   |
| twwtuititi | BOOL      | <code>GopherGetAttributeW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszLocator, _In_ LPCWSTR lpszAttributeName, _Out_ LPBYTE lpBuffer, _In_ DWORD dwBufferLength, _Out_ LPDWORD lpdwCharactersReturned, _In_ GOPHER_ATTRIBUTE_ENUMERATOR lpfmEnumerator, _In_ DWORD_PTR dwContext)</code> |
| sti        | BOOL      | <code>GopherGetLocatorType(_In_ LPCTSTR lpszLocator, _Out_ LPDWORD lpdwGopherType)</code>                                                                                                                                                                                              |
|            |           | <code>GopherGetLocatorTypeA(_In_ LPCSTR lpszLocator, _Out_</code>                                                                                                                                                                                                                      |

|            |           |                                                                                                                                                                                                                                  |
|------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ati        | BOOL      | LPDWORD lpdwGopherType)                                                                                                                                                                                                          |
| wti        | BOOL      | GopherGetLocatorTypeW(_In_ LPCWSTR lpszLocator, _Out_ LPDWORD lpdwGopherType)                                                                                                                                                    |
| tssuitt    | HINTERNET | GopherOpenFile(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszLocator, _In_ LPCTSTR lpszView, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                                                                                           |
| taauitt    | HINTERNET | GopherOpenFileA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszLocator, _In_ LPCSTR lpszView, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                                                                                            |
| twwuitt    | HINTERNET | GopherOpenFileW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszLocator, _In_ LPCWSTR lpszView, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                                                                                          |
| tsuiiii    | BOOL      | HttpAddRequestHeaders(_In_ HINTERNET hRequest, _In_ LPCTSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwModifiers)                                                                                                     |
| tauiiii    | BOOL      | HttpAddRequestHeadersA(_In_ HINTERNET hRequest, _In_ LPCSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwModifiers)                                                                                                     |
| twuiiii    | BOOL      | HttpAddRequestHeadersW(_In_ HINTERNET hRequest, _In_ LPCWSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwModifiers)                                                                                                    |
| ttuiti     | BOOL      | HttpEndRequest(_In_ HINTERNET hRequest, _Out_opt_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_opt_ DWORD_PTR dwContext)                                                                                             |
| ttuiti     | BOOL      | HttpEndRequestA(_In_ HINTERNET hRequest, _Out_opt_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_opt_ DWORD_PTR dwContext)                                                                                            |
| ttuiti     | BOOL      | HttpEndRequestW(_In_ HINTERNET hRequest, _Out_opt_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_opt_ DWORD_PTR dwContext)                                                                                            |
| tssssuitt  | HINTERNET | HttpOpenRequest(_In_ HINTERNET hConnect, _In_ LPCTSTR lpszVerb, _In_ LPCTSTR lpszObjectName, _In_ LPCTSTR lpszVersion, _In_ LPCTSTR lpszReferer, _In_ LPCTSTR *lplpszAcceptTypes, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)  |
| taaaaauitt | HINTERNET | HttpOpenRequestA(_In_ HINTERNET hConnect, _In_ LPCSTR lpszVerb, _In_ LPCSTR lpszObjectName, _In_ LPCSTR lpszVersion, _In_ LPCSTR lpszReferer, _In_ LPCSTR *lplpszAcceptTypes, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)      |
| twwwwwuitt | HINTERNET | HttpOpenRequestW(_In_ HINTERNET hConnect, _In_ LPCWSTR lpszVerb, _In_ LPCWSTR lpszObjectName, _In_ LPCWSTR lpszVersion, _In_ LPCWSTR lpszReferer, _In_ LPCWSTR *lplpszAcceptTypes, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext) |

|          |       |                                                                                                                                                                                        |
|----------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitti   | BOOL  | <a href="#">HttpQueryInfo</a> (_In_ HINTERNET hRequest, _In_ DWORD dwInfoLevel, _Inout_ LPVOID lpvBuffer, _Inout_ LPDWORD lpdwBufferLength, _Inout_ LPDWORD lpdwIndex)                 |
| tuitti   | BOOL  | <a href="#">HttpQueryInfoA</a> (_In_ HINTERNET hRequest, _In_ DWORD dwInfoLevel, _Inout_ LPVOID lpvBuffer, _Inout_ LPDWORD lpdwBufferLength, _Inout_ LPDWORD lpdwIndex)                |
| tuitti   | BOOL  | <a href="#">HttpQueryInfoW</a> (_In_ HINTERNET hRequest, _In_ DWORD dwInfoLevel, _Inout_ LPVOID lpvBuffer, _Inout_ LPDWORD lpdwBufferLength, _Inout_ LPDWORD lpdwIndex)                |
| tsuituii | BOOL  | <a href="#">HttpSendRequest</a> (_In_ HINTERNET hRequest, _In_ LPCTSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ LPVOID lpOptional, _In_ DWORD dwOptionalLength)                   |
| tauituii | BOOL  | <a href="#">HttpSendRequestA</a> (_In_ HINTERNET hRequest, _In_ LPCSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ LPVOID lpOptional, _In_ DWORD dwOptionalLength)                   |
| ttuiti   | BOOL  | <a href="#">HttpSendRequestEx</a> (_In_ HINTERNET hRequest, _In_ LPINTERNET_BUFFERS lpBuffersIn, _Out_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)  |
| ttuiti   | BOOL  | <a href="#">HttpSendRequestExA</a> (_In_ HINTERNET hRequest, _In_ LPINTERNET_BUFFERS lpBuffersIn, _Out_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext) |
| ttuiti   | BOOL  | <a href="#">HttpSendRequestExW</a> (_In_ HINTERNET hRequest, _In_ LPINTERNET_BUFFERS lpBuffersIn, _Out_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext) |
| twuituii | BOOL  | <a href="#">HttpSendRequestW</a> (_In_ HINTERNET hRequest, _In_ LPCWSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ LPVOID lpOptional, _In_ DWORD dwOptionalLength)                  |
| uiui     | DWORD | <a href="#">InternetAttemptConnect</a> (_In_ DWORD dwReserved)                                                                                                                         |
| uiti     | BOOL  | <a href="#">InternetAutodial</a> (_In_ DWORD dwFlags, _In_ HWND hwndParent)                                                                                                            |
| uii      | BOOL  | <a href="#">InternetAutodialHangup</a> (_In_ DWORD dwReserved)                                                                                                                         |
| sstuii   | BOOL  | <a href="#">InternetCanonicalizeUrl</a> (_In_ LPCTSTR lpszUrl, _Out_ LPTSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength, _In_ DWORD dwFlags)                                          |
| aatuii   | BOOL  | <a href="#">InternetCanonicalizeUrlA</a> (_In_ LPCSTR lpszUrl, _Out_ LPSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength,                                                               |

|              |           |                                                                                                                                                                                                                                  |
|--------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              |           | _In_ DWORD dwFlags)                                                                                                                                                                                                              |
| wwtiii       | BOOL      | InternetCanonicalizeUrlW(_In_ LPCWSTR lpszUrl, _Out_ LPWSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength, _In_ DWORD dwFlags)                                                                                                    |
| suiiii       | BOOL      | InternetCheckConnection(_In_ LPCTSTR lpszUrl, _In_ DWORD dwFlags, _In_ DWORD dwReserved)                                                                                                                                         |
| aiuiii       | BOOL      | InternetCheckConnectionA(_In_ LPCSTR lpszUrl, _In_ DWORD dwFlags, _In_ DWORD dwReserved)                                                                                                                                         |
| wuiiii       | BOOL      | InternetCheckConnectionW(_In_ LPCWSTR lpszUrl, _In_ DWORD dwFlags, _In_ DWORD dwReserved)                                                                                                                                        |
| i            | BOOL      | InternetClearAllPerSiteCookieDecisions(void)                                                                                                                                                                                     |
| ti           | BOOL      | InternetCloseHandle(_In_ HINTERNET hInternet)                                                                                                                                                                                    |
| ssstiii      | BOOL      | InternetCombineUrl(_In_ LPCTSTR lpszBaseUrl, _In_ LPCTSTR lpszRelativeUrl, _Out_ LPTSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength, _In_ DWORD dwFlags)                                                                        |
| aaatiii      | BOOL      | InternetCombineUrlA(_In_ LPCSTR lpszBaseUrl, _In_ LPCSTR lpszRelativeUrl, _Out_ LPSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength, _In_ DWORD dwFlags)                                                                          |
| wwwtiii      | BOOL      | InternetCombineUrlW(_In_ LPCWSTR lpszBaseUrl, _In_ LPCWSTR lpszRelativeUrl, _Out_ LPWSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength, _In_ DWORD dwFlags)                                                                       |
| tssiii       | DWORD     | InternetConfirmZoneCrossing(_In_ HWND hWnd, _In_ LPTSTR szUrlPrev, _In_ LPTSTR szUrlNew, _In_ BOOL bPost)                                                                                                                        |
| taaiiii      | DWORD     | InternetConfirmZoneCrossingA(_In_ HWND hWnd, _In_ LPSTR szUrlPrev, _In_ LPSTR szUrlNew, _In_ BOOL bPost)                                                                                                                         |
| twwiiii      | DWORD     | InternetConfirmZoneCrossingW(_In_ HWND hWnd, _In_ LPWSTR szUrlPrev, _In_ LPWSTR szUrlNew, _In_ BOOL bPost)                                                                                                                       |
| tsuhssuiiitt | HINTERNET | InternetConnect(_In_ HINTERNET hInternet, _In_ LPCTSTR lpszServerName, _In_ INTERNET_PORT nServerPort, _In_ LPCTSTR lpszUsername, _In_ LPCTSTR lpszPassword, _In_ DWORD dwService, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext) |
| tauhaauiiitt | HINTERNET | InternetConnectA(_In_ HINTERNET hInternet, _In_ LPCSTR lpszServerName, _In_ INTERNET_PORT nServerPort, _In_ LPCSTR lpszUsername, _In_ LPCSTR lpszPassword, _In_ DWORD dwService, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)   |
|              |           | InternetConnectW(_In_ HINTERNET hInternet, _In_ LPCWSTR lpszServerName, _In_ INTERNET_PORT                                                                                                                                       |

|             |           |                                                                                                                                                                   |
|-------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| twuhwwuiitt | HINTERNET | nServerPort, _In_ LPCWSTR lpszUsername, _In_ LPCWSTR lpszPassword, _In_ DWORD dwService, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                            |
| suiiiti     | BOOL      | InternetCrackUrl(_In_ LPCTSTR lpszUrl, _In_ DWORD dwUrlLength, _In_ DWORD dwFlags, _Inout_ LPURL_COMPONENTS lpUrlComponents)                                      |
| aiiiti      | BOOL      | InternetCrackUrlA(_In_ LPCSTR lpszUrl, _In_ DWORD dwUrlLength, _In_ DWORD dwFlags, _Inout_ LPURL_COMPONENTS lpUrlComponents)                                      |
| wuiiiti     | BOOL      | InternetCrackUrlW(_In_ LPCWSTR lpszUrl, _In_ DWORD dwUrlLength, _In_ DWORD dwFlags, _Inout_ LPURL_COMPONENTS lpUrlComponents)                                     |
| tuiiti      | BOOL      | InternetCreateUrl(_In_ LPURL_COMPONENTS lpUrlComponents, _In_ DWORD dwFlags, _Out_ LPTSTR lpszUrl, _Inout_ LPDWORD lpdwUrlLength)                                 |
| tuiati      | BOOL      | InternetCreateUrlA(_In_ LPURL_COMPONENTS lpUrlComponents, _In_ DWORD dwFlags, _Out_ LPSTR lpszUrl, _Inout_ LPDWORD lpdwUrlLength)                                 |
| tuiwiti     | BOOL      | InternetCreateUrlW(_In_ LPURL_COMPONENTS lpUrlComponents, _In_ DWORD dwFlags, _Out_ LPWSTR lpszUrl, _Inout_ LPDWORD lpdwUrlLength)                                |
| tsuituii    | DWORD     | InternetDial(_In_ HWND hwndParent, _In_ LPTSTR pszEntryName, _In_ DWORD dwFlags, _Out_ DWORD_PTR *lpdwConnection, _In_ DWORD dwReserved)                          |
| tauituii    | DWORD     | InternetDialA(_In_ HWND hwndParent, _In_ LPSTR pszEntryName, _In_ DWORD dwFlags, _Out_ DWORD_PTR *lpdwConnection, _In_ DWORD dwReserved)                          |
| twuituii    | DWORD     | InternetDialW(_In_ HWND hwndParent, _In_ LPWSTR pszEntryName, _In_ DWORD dwFlags, _Out_ DWORD_PTR *lpdwConnection, _In_ DWORD dwReserved)                         |
| sttiii      | BOOL      | InternetEnumPerSiteCookieDecision(_Out_ LPTSTR pszSiteName, _Inout_ unsigned long *pcSiteNameSize, _Out_ unsigned long *pdwDecision, _In_ unsigned long dwIndex)  |
| attiii      | BOOL      | InternetEnumPerSiteCookieDecisionA(_Out_ LPSTR pszSiteName, _Inout_ unsigned long *pcSiteNameSize, _Out_ unsigned long *pdwDecision, _In_ unsigned long dwIndex)  |
| wttiii      | BOOL      | InternetEnumPerSiteCookieDecisionW(_Out_ LPWSTR pszSiteName, _Inout_ unsigned long *pcSiteNameSize, _Out_ unsigned long *pdwDecision, _In_ unsigned long dwIndex) |
|             |           |                                                                                                                                                                   |

|           |       |                                                                                                                                                                                  |
|-----------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiuitui | DWORD | InternetErrorDlg(_In_ HWND hWnd, _Inout_ HINTERNET hRequest, _In_ DWORD dwError, _In_ DWORD dwFlags, _Inout_ LPVOID *lppvData)                                                   |
| tti       | BOOL  | InternetFindNextFile(_In_ HINTERNET hFind, _Out_ LPVOID lpvFindData)                                                                                                             |
| tti       | BOOL  | InternetFindNextFileA(_In_ HINTERNET hFind, _Out_ LPVOID lpvFindData)                                                                                                            |
| tti       | BOOL  | InternetFindNextFileW(_In_ HINTERNET hFind, _Out_ LPVOID lpvFindData)                                                                                                            |
| tuii      | BOOL  | InternetGetConnectedState(_Out_ LPDWORD lpdwFlags, _In_ DWORD dwReserved)                                                                                                        |
| tsuiuii   | BOOL  | InternetGetConnectedStateEx(_Out_ LPDWORD lpdwFlags, _Out_ LPTSTR lpszConnectionName, _In_ DWORD dwNameLen, _In_ DWORD dwReserved)                                               |
| tauiuii   | BOOL  | InternetGetConnectedStateExA(_Out_ LPDWORD lpdwFlags, _Out_ LPSTR lpszConnectionName, _In_ DWORD dwNameLen, _In_ DWORD dwReserved)                                               |
| twuiuii   | BOOL  | InternetGetConnectedStateExW(_Out_ LPDWORD lpdwFlags, _Out_ LPWSTR lpszConnectionName, _In_ DWORD dwNameLen, _In_ DWORD dwReserved)                                              |
| sssti     | BOOL  | InternetGetCookie(_In_ LPCTSTR lpszUrl, _In_ LPCTSTR lpszCookieName, _Out_ LPTSTR lpszCookieData, _Inout_ LPDWORD lpdwSize)                                                      |
| aaati     | BOOL  | InternetGetCookieA(_In_ LPCSTR lpszUrl, _In_ LPCSTR lpszCookieName, _Out_ LPSTR lpszCookieData, _Inout_ LPDWORD lpdwSize)                                                        |
| ssstuiti  | BOOL  | InternetGetCookieEx(_In_ LPCTSTR lpszURL, _In_ LPCTSTR lpszCookieName, _Inout_opt_ LPTSTR lpszCookieData, _Inout_ LPDWORD lpdwSize, _In_ DWORD dwFlags, _In_ LPVOID lpReserved)  |
| aaatuiti  | BOOL  | InternetGetCookieExA(_In_ LPCSTR lpszURL, _In_ LPCSTR lpszCookieName, _Inout_opt_ LPSTR lpszCookieData, _Inout_ LPDWORD lpdwSize, _In_ DWORD dwFlags, _In_ LPVOID lpReserved)    |
| wwwtuiti  | BOOL  | InternetGetCookieExW(_In_ LPCWSTR lpszURL, _In_ LPCWSTR lpszCookieName, _Inout_opt_ LPWSTR lpszCookieData, _Inout_ LPDWORD lpdwSize, _In_ DWORD dwFlags, _In_ LPVOID lpReserved) |
| wwwti     | BOOL  | InternetGetCookieW(_In_ LPCWSTR lpszUrl, _In_ LPCWSTR lpszCookieName, _Out_ LPWSTR lpszCookieData, _Inout_ LPDWORD lpdwSize)                                                     |
| tsti      | BOOL  | InternetGetLastResponseInfo(_Out_ LPDWORD lpdwError, _Out_ LPTSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                                  |
|           |       |                                                                                                                                                                                  |

|           |           |                                                                                                                                                                                       |
|-----------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tati      | BOOL      | <a href="#">InternetGetLastResponseInfoA</a> (_Out_ LPDWORD lpdwError, _Out_ LPSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                      |
| twti      | BOOL      | <a href="#">InternetGetLastResponseInfoW</a> (_Out_ LPDWORD lpdwError, _Out_ LPWSTR lpszBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                     |
| sti       | BOOL      | <a href="#">InternetGetPerSiteCookieDecision</a> (_In_ LPCTSTR pchHostName, _Out_ unsigned long *pResult)                                                                             |
| ati       | BOOL      | <a href="#">InternetGetPerSiteCookieDecisionA</a> (_In_ LPCSTR pchHostName, _Out_ unsigned long *pResult)                                                                             |
| wti       | BOOL      | <a href="#">InternetGetPerSiteCookieDecisionW</a> (_In_ LPCWSTR pchHostName, _Out_ unsigned long *pResult)                                                                            |
| stuii     | BOOL      | <a href="#">InternetGoOnline</a> (_In_ LPTSTR lpszURL, _In_ HWND hwndParent, _In_ DWORD dwFlags)                                                                                      |
| atuii     | BOOL      | <a href="#">InternetGoOnlineA</a> (_In_ LPSTR lpszURL, _In_ HWND hwndParent, _In_ DWORD dwFlags)                                                                                      |
| wtuii     | BOOL      | <a href="#">InternetGoOnlineW</a> (_In_ LPWSTR lpszURL, _In_ HWND hwndParent, _In_ DWORD dwFlags)                                                                                     |
| tuiui     | DWORD     | <a href="#">InternetHangUp</a> (_In_ DWORD_PTR dwConnection, _In_ DWORD dwReserved)                                                                                                   |
| tii       | BOOL      | <a href="#">InternetLockRequestFile</a> (_In_ HINTERNET hInternet, _Out_ HANDLE *lphLockReqHandle)                                                                                    |
| suissuit  | HINTERNET | <a href="#">InternetOpen</a> (_In_ LPCTSTR lpszAgent, _In_ DWORD dwAccessType, _In_ LPCTSTR lpszProxyName, _In_ LPCTSTR lpszProxyBypass, _In_ DWORD dwFlags)                          |
| auiaauit  | HINTERNET | <a href="#">InternetOpenA</a> (_In_ LPCSTR lpszAgent, _In_ DWORD dwAccessType, _In_ LPCSTR lpszProxyName, _In_ LPCSTR lpszProxyBypass, _In_ DWORD dwFlags)                            |
| tssuiuit  | HINTERNET | <a href="#">InternetOpenUrl</a> (_In_ HINTERNET hInternet, _In_ LPCTSTR lpszUrl, _In_ LPCTSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)  |
| taauiuitt | HINTERNET | <a href="#">InternetOpenUrlA</a> (_In_ HINTERNET hInternet, _In_ LPCSTR lpszUrl, _In_ LPCSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)   |
| twwuiuit  | HINTERNET | <a href="#">InternetOpenUrlW</a> (_In_ HINTERNET hInternet, _In_ LPCWSTR lpszUrl, _In_ LPCWSTR lpszHeaders, _In_ DWORD dwHeadersLength, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext) |
| wuiwwuit  | HINTERNET | <a href="#">InternetOpenW</a> (_In_ LPCWSTR lpszAgent, _In_ DWORD dwAccessType, _In_ LPCWSTR lpszProxyName, _In_ LPCWSTR lpszProxyBypass, _In_ DWORD dwFlags)                         |
|           |           |                                                                                                                                                                                       |

|           |       |                                                                                                                                                                      |
|-----------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiti    | BOOL  | <a href="#">InternetQueryDataAvailable</a> (_In_ HINTERNET hFile, _Out_ LPDWORD lpdwNumberOfBytesAvailable, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)            |
| tuitti    | BOOL  | <a href="#">InternetQueryOption</a> (_In_ HINTERNET hInternet, _In_ DWORD dwOption, _Out_ LPVOID lpBuffer, _Inout_ LPDWORD lpdwBufferLength)                         |
| tuitti    | BOOL  | <a href="#">InternetQueryOptionA</a> (_In_ HINTERNET hInternet, _In_ DWORD dwOption, _Out_ LPVOID lpBuffer, _Inout_ LPDWORD lpdwBufferLength)                        |
| tuitti    | BOOL  | <a href="#">InternetQueryOptionW</a> (_In_ HINTERNET hInternet, _In_ DWORD dwOption, _Out_ LPVOID lpBuffer, _Inout_ LPDWORD lpdwBufferLength)                        |
| ttuiti    | BOOL  | <a href="#">InternetReadFile</a> (_In_ HINTERNET hFile, _Out_ LPVOID lpBuffer, _In_ DWORD dwNumberOfBytesToRead, _Out_ LPDWORD lpdwNumberOfBytesRead)                |
| ttuiti    | BOOL  | <a href="#">InternetReadFileEx</a> (_In_ HINTERNET hFile, _Out_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                       |
| ttuiti    | BOOL  | <a href="#">InternetReadFileExA</a> (_In_ HINTERNET hFile, _Out_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                      |
| ttuiti    | BOOL  | <a href="#">InternetReadFileExW</a> (_In_ HINTERNET hFile, _Out_ LPINTERNET_BUFFERS lpBuffersOut, _In_ DWORD dwFlags, _In_ DWORD_PTR dwContext)                      |
| sssi      | BOOL  | <a href="#">InternetSetCookie</a> (_In_ LPCTSTR lpszUrl, _In_ LPCTSTR lpszCookieName, _In_ LPCTSTR lpszCookieData)                                                   |
| aaai      | BOOL  | <a href="#">InternetSetCookieA</a> (_In_ LPCSTR lpszUrl, _In_ LPCSTR lpszCookieName, _In_ LPCSTR lpszCookieData)                                                     |
| sssuitui  | DWORD | <a href="#">InternetSetCookieEx</a> (_In_ LPCTSTR lpszURL, _In_ LPCTSTR lpszCookieName, _In_ LPCTSTR lpszCookieData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwReserved)  |
| aaauitui  | DWORD | <a href="#">InternetSetCookieExA</a> (_In_ LPCSTR lpszURL, _In_ LPCSTR lpszCookieName, _In_ LPCSTR lpszCookieData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwReserved)    |
| wwwuitui  | DWORD | <a href="#">InternetSetCookieExW</a> (_In_ LPCWSTR lpszURL, _In_ LPCWSTR lpszCookieName, _In_ LPCWSTR lpszCookieData, _In_ DWORD dwFlags, _In_ DWORD_PTR dwReserved) |
| wwwi      | BOOL  | <a href="#">InternetSetCookieW</a> (_In_ LPCWSTR lpszUrl, _In_ LPCWSTR lpszCookieName, _In_ LPCWSTR lpszCookieData)                                                  |
| tuituitui | DWORD | <a href="#">InternetSetFilePointer</a> (_In_ HINTERNET hFile, _In_ LONG lDistanceToMove, _Inout_ PLONG lpDistanceToMoveHigh, _In_ DWORD dwMoveMethod,                |

|         |                  |                                                                                                                     |
|---------|------------------|---------------------------------------------------------------------------------------------------------------------|
|         |                  | _In_ DWORD_PTR dwContext)                                                                                           |
| tuituii | BOOL             | InternetSetOption(_In_ HINTERNET hInternet, _In_ DWORD dwOption, _In_ LPVOID lpBuffer, _In_ DWORD dwBufferLength)   |
| tuituii | BOOL             | InternetSetOptionA(_In_ HINTERNET hInternet, _In_ DWORD dwOption, _In_ LPVOID lpBuffer, _In_ DWORD dwBufferLength)  |
| tuituii | BOOL             | InternetSetOptionW(_In_ HINTERNET hInternet, _In_ DWORD dwOption, _In_ LPVOID lpBuffer, _In_ DWORD dwBufferLength)  |
| suii    | BOOL             | InternetSetPerSiteCookieDecision(_In_ LPCTSTR pchHostName, _In_ DWORD dwDecision)                                   |
| auui    | BOOL             | InternetSetPerSiteCookieDecisionA(_In_ LPCSTR pchHostName, _In_ DWORD dwDecision)                                   |
| wuui    | BOOL             | InternetSetPerSiteCookieDecisionW(_In_ LPCWSTR pchHostName, _In_ DWORD dwDecision)                                  |
| ttt     | INTERNET_STATUS_ | InternetSetStatusCallback(_In_ HINTERNET hInternet, _In_ INTERNET_STATUS_CALLBACK lpfnInternetCallback)             |
| ttt     | INTERNET_STATUS_ | InternetSetStatusCallbackA(_In_ HINTERNET hInternet, _In_ INTERNET_STATUS_CALLBACK lpfnInternetCallback)            |
| ttt     | INTERNET_STATUS_ | InternetSetStatusCallbackW(_In_ HINTERNET hInternet, _In_ INTERNET_STATUS_CALLBACK lpfnInternetCallback)            |
| tuisuii | BOOL             | InternetTimeFromSystemTime(_In_ const SYSTEMTIME *pst, _In_ DWORD dwRFC, _Out_ LPTSTR lpszTime, _In_ DWORD cbTime)  |
| tuiauui | BOOL             | InternetTimeFromSystemTimeA(_In_ const SYSTEMTIME *pst, _In_ DWORD dwRFC, _Out_ LPSTR lpszTime, _In_ DWORD cbTime)  |
| tuiwuui | BOOL             | InternetTimeFromSystemTimeW(_In_ const SYSTEMTIME *pst, _In_ DWORD dwRFC, _Out_ LPWSTR lpszTime, _In_ DWORD cbTime) |
| stuii   | BOOL             | InternetTimeToSystemTime(_In_ LPCTSTR lpszTime, _Out_ SYSTEMTIME *pst, _In_ DWORD dwReserved)                       |
| atuii   | BOOL             | InternetTimeToSystemTimeA(_In_ LPCSTR lpszTime, _Out_ SYSTEMTIME *pst, _In_ DWORD dwReserved)                       |
| wtuii   | BOOL             | InternetTimeToSystemTimeW(_In_ LPCWSTR lpszTime, _Out_ SYSTEMTIME *pst, _In_ DWORD dwReserved)                      |
| ti      | BOOL             | InternetUnlockRequestFile(_In_ HANDLE hLockRequestInfo)                                                             |
|         |                  | InternetWriteFile(_In_ HINTERNET hFile, _In_ LPCVOID                                                                |

|            |        |                                                                                                                                                                                              |
|------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ttuiti     | BOOL   | lpBuffer, _In_ DWORD dwNumberOfBytesToWrite, _Out_ LPDWORD lpdwNumberOfBytesWritten)                                                                                                         |
| uiuitttui  | DWORD  | PrivacyGetZonePreferenceW(_In_ DWORD dwZone, _In_ DWORD dwType, _Out_opt_ LPDWORD pdwTemplate, _Out_opt_ LPWSTR *pszBuffer, _Inout_opt_ LPDWORD pdwBufferLength)                             |
| uiuiuiwui  | DWORD  | PrivacySetZonePreferenceW(_In_ DWORD dwZone, _In_ DWORD dwType, _In_ DWORD dwTemplate, _In_opt_ LPCWSTR pszPreference)                                                                       |
| tuittui    | BOOL   | ReadUrlCacheEntryStream(_In_ HANDLE hUrlCacheStream, _In_ DWORD dwLocation, _Inout_ LPVOID lpBuffer, _Inout_ LPDWORD lpdwLen, _In_ DWORD dwReserved)                                         |
| tuii       | BOOL   | ResumeSuspendedDownload(_In_ HINTERNET hRequest, _In_ DWORD dwResultCode)                                                                                                                    |
| sttuii     | BOOL   | RetrieveUrlCacheEntryFile(_In_ LPCTSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo, _In_ DWORD dwReserved)                           |
| attuii     | BOOL   | RetrieveUrlCacheEntryFileA(_In_ LPCSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo, _In_ DWORD dwReserved)                           |
| wttuii     | BOOL   | RetrieveUrlCacheEntryFileW(_In_ LPCWSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo, _In_ DWORD dwReserved)                          |
| sttuit     | HANDLE | RetrieveUrlCacheEntryStream(_In_ LPCTSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo, _In_ BOOL fRandomRead, _In_ DWORD dwReserved)  |
| attuit     | HANDLE | RetrieveUrlCacheEntryStreamA(_In_ LPCSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo, _In_ BOOL fRandomRead, _In_ DWORD dwReserved)  |
| wttuit     | HANDLE | RetrieveUrlCacheEntryStreamW(_In_ LPCWSTR lpszUrlName, _Out_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _Inout_ LPDWORD lpcbCacheEntryInfo, _In_ BOOL fRandomRead, _In_ DWORD dwReserved) |
| suii6tuiti | BOOL   | SetUrlCacheEntryGroup(_In_ LPCTSTR lpszUrlName, _In_ DWORD dwFlags, _In_ GROUPID GroupId, _In_ LPBYTE pbGroupAttributes, _In_ DWORD                                                          |

|            |      |                                                                                                                                                                                 |
|------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |      | cbGroupAttributes, _In_ LPVOID lpReserved)                                                                                                                                      |
| auii6tuiti | BOOL | SetUrlCacheEntryGroupA(_In_ LPCSTR lpszUrlName, _In_ DWORD dwFlags, _In_ GROUPID GroupId, _In_ LPBYTE pbGroupAttributes, _In_ DWORD cbGroupAttributes, _In_ LPVOID lpReserved)  |
| wuii6tuiti | BOOL | SetUrlCacheEntryGroupW(_In_ LPCWSTR lpszUrlName, _In_ DWORD dwFlags, _In_ GROUPID GroupId, _In_ LPBYTE pbGroupAttributes, _In_ DWORD cbGroupAttributes, _In_ LPVOID lpReserved) |
| stuii      | BOOL | SetUrlCacheEntryInfo(_In_ LPCTSTR lpszUrlName, _In_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _In_ DWORD dwFieldControl)                                                    |
| atuii      | BOOL | SetUrlCacheEntryInfoA(_In_ LPCSTR lpszUrlName, _In_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _In_ DWORD dwFieldControl)                                                    |
| wtuii      | BOOL | SetUrlCacheEntryInfoW(_In_ LPCWSTR lpszUrlName, _In_ LPINTERNET_CACHE_ENTRY_INFO lpCacheEntryInfo, _In_ DWORD dwFieldControl)                                                   |
| i6uiiuitti | BOOL | SetUrlCacheGroupAttribute(_In_ GROUPID gid, _In_ DWORD dwFlags, _In_ DWORD dwAttributes, _In_ LPINTERNET_CACHE_GROUP_INFO lpGroupInfo, _Inout_ LPVOID lpReserved)               |
| i6uiiuitti | BOOL | SetUrlCacheGroupAttributeA(_In_ GROUPID gid, _In_ DWORD dwFlags, _In_ DWORD dwAttributes, _In_ LPINTERNET_CACHE_GROUP_INFO lpGroupInfo, _Inout_ LPVOID lpReserved)              |
| i6uiiuitti | BOOL | SetUrlCacheGroupAttributeW(_In_ GROUPID gid, _In_ DWORD dwFlags, _In_ DWORD dwAttributes, _In_ LPINTERNET_CACHE_GROUP_INFO lpGroupInfo, _Inout_ LPVOID lpReserved)              |
| suii       | BOOL | UnlockUrlCacheEntryFile(_In_ LPCTSTR lpszUrlName, _In_ DWORD dwReserved)                                                                                                        |
| auii       | BOOL | UnlockUrlCacheEntryFileA(_In_ LPCSTR lpszUrlName, _In_ DWORD dwReserved)                                                                                                        |
| wuii       | BOOL | UnlockUrlCacheEntryFileW(_In_ LPCWSTR lpszUrlName, _In_ DWORD dwReserved)                                                                                                       |
| tuii       | BOOL | UnlockUrlCacheEntryStream(_In_ HANDLE hUrlCacheStream, _In_ DWORD dwReserved)                                                                                                   |

# Winmm.dll

|                |          |                                                                                                                                     |
|----------------|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| ttuiui         | MMRESULT | <code>auxGetDevCaps</code> (UINT_PTR uDeviceID, LPAUXCAPS lpCaps, UINT cbCaps)                                                      |
| ttuiui         | MMRESULT | <code>auxGetDevCapsA</code> (UINT_PTR uDeviceID, LPAUXCAPS lpCaps, UINT cbCaps)                                                     |
| ttuiui         | MMRESULT | <code>auxGetDevCapsW</code> (UINT_PTR uDeviceID, LPAUXCAPS lpCaps, UINT cbCaps)                                                     |
| ui             | UINT     | <code>auxGetNumDevs</code> (void)                                                                                                   |
| uitui          | MMRESULT | <code>auxGetVolume</code> (UINT uDeviceID, LPDWORD lpdwVolume)                                                                      |
| uiuittui       | DWORD    | <code>auxGetMessage</code> (UINT uDeviceID, UINT uMsg, DWORD_PTR dwParam1, DWORD_PTR dwParam2)                                      |
| uiuiui         | MMRESULT | <code>auxSetVolume</code> (UINT uDeviceID, DWORD dwVolume)                                                                          |
| tttt           | LRESULT  | <code>CloseDriver</code> (_In_ HDRVR hdrvr, _In_ LPARAM lParam1, _In_ LPARAM lParam2)                                               |
| ttuiuiuit      | LRESULT  | <code>DefDriverProc</code> (DWORD_PTR dwDriverId, HDRVR hdrvr, UINT msg, LONG lParam1, LONG lParam2)                                |
| uiuituiuiuiiii | BOOL     | <code>DriverCallback</code> (DWORD dwCallBack, DWORD dwFlags, HDRVR hdrvr, DWORD msg, DWORD dwUser, DWORD dwParam1, DWORD dwParam2) |
| tt             | HMODULE  | <code>DrvGetModuleHandle</code> (_In_ HDRVR hDriver)                                                                                |
| tt             | HMODULE  | <code>GetDriverModuleHandle</code> (_In_ HDRVR hdrvr)                                                                               |
| uiui           | MMRESULT | <code>joyConfigChanged</code> (DWORD dwFlags)                                                                                       |
| ttuiui         | MMRESULT | <code>joyGetDevCaps</code> (UINT_PTR uJoyID, LPJOYCAPS pjc, UINT cbjc)                                                              |
| ttuiui         | MMRESULT | <code>joyGetDevCapsA</code> (UINT_PTR uJoyID, LPJOYCAPS pjc, UINT cbjc)                                                             |
| ttuiui         | MMRESULT | <code>joyGetDevCapsW</code> (UINT_PTR uJoyID, LPJOYCAPS pjc, UINT cbjc)                                                             |
| ui             | UINT     | <code>joyGetNumDevs</code> (void)                                                                                                   |
| uitui          | MMRESULT | <code>joyGetPos</code> (UINT uJoyID, LPJOYINFO pji)                                                                                 |
| uitui          | MMRESULT | <code>joyGetPosEx</code> (UINT uJoyID, LPJOYINFOEX pji)                                                                             |
| uitui          | MMRESULT | <code>joyGetThreshold</code> (UINT uJoyID, LPUINT puThreshold)                                                                      |
| uiui           | MMRESULT | <code>joyReleaseCapture</code> (UINT uJoyID)                                                                                        |
| tuiuiui        | MMRESULT | <code>joySetCapture</code> (HWND hwnd, UINT uJoyID, UINT uPeriod, BOOL fChanged)                                                    |
| uiuiui         | MMRESULT | <code>joySetThreshold</code> (UINT uJoyID, UINT uThreshold)                                                                         |
| ai             | BOOL     | <code>nciExecute</code> (LPCSTR pszCommand)                                                                                         |
|                |          |                                                                                                                                     |

|          |             |                                                                                                            |
|----------|-------------|------------------------------------------------------------------------------------------------------------|
| uit      | HANDLE      | <i>mcidGetCreatorTask</i> (MCIDEVICEID IDDevice)                                                           |
| sui      | MCIDEVICEID | <i>mcidGetDeviceID</i> (LPCTSTR lpszDevice)                                                                |
| au       | MCIDEVICEID | <i>mcidGetDeviceIDA</i> (LPCSTR lpszDevice)                                                                |
| uisui    | MCIDEVICEID | <i>mcidGetDeviceIDFromElementID</i> (DWORD dwElementID, LPCTSTR lpstrType)                                 |
| uiaui    | MCIDEVICEID | <i>mcidGetDeviceIDFromElementIDA</i> (DWORD dwElementID, LPCSTR lpstrType)                                 |
| uiwui    | MCIDEVICEID | <i>mcidGetDeviceIDFromElementIDW</i> (DWORD dwElementID, LPCWSTR lpstrType)                                |
| wui      | MCIDEVICEID | <i>mcidGetDeviceIDW</i> (LPCWSTR lpszDevice)                                                               |
| uisuii   | BOOL        | <i>mcidGetErrorString</i> (DWORD fdwError, LPTSTR lpszErrorText, UINT cchErrorText)                        |
| uiauii   | BOOL        | <i>mcidGetErrorStringA</i> (DWORD fdwError, LPSTR lpszErrorText, UINT cchErrorText)                        |
| uiwuii   | BOOL        | <i>mcidGetErrorStringW</i> (DWORD fdwError, LPWSTR lpszErrorText, UINT cchErrorText)                       |
| uitt     | YIELDPROC   | <i>mcidGetYieldProc</i> (MCIDEVICEID IDDevice, LPDWORD lpdwYieldData)                                      |
| uiuittui | MCIERROR    | <i>mcidSendCommand</i> (MCIDEVICEID IDDevice, UINT uMsg, DWORD_PTR fdwCommand, DWORD_PTR dwParam)          |
| uiuittui | MCIERROR    | <i>mcidSendCommandA</i> (MCIDEVICEID IDDevice, UINT uMsg, DWORD_PTR fdwCommand, DWORD_PTR dwParam)         |
| uiuittui | MCIERROR    | <i>mcidSendCommandW</i> (MCIDEVICEID IDDevice, UINT uMsg, DWORD_PTR fdwCommand, DWORD_PTR dwParam)         |
| ssuitui  | MCIERROR    | <i>mcidSendString</i> (LPCTSTR lpszCommand, LPTSTR lpszReturnString, UINT cchReturn, HANDLE hwndCallback)  |
| aaui     | MCIERROR    | <i>mcidSendStringA</i> (LPCSTR lpszCommand, LPSTR lpszReturnString, UINT cchReturn, HANDLE hwndCallback)   |
| wwuitui  | MCIERROR    | <i>mcidSendStringW</i> (LPCWSTR lpszCommand, LPWSTR lpszReturnString, UINT cchReturn, HANDLE hwndCallback) |
| uituiui  | UINT        | <i>mcidSetYieldProc</i> (MCIDEVICEID IDDevice, YIELDPROC yp, DWORD dwYieldData)                            |
| ttui     | MMRESULT    | <i>midicConnect</i> (HMIDI hMidi, HMIDIOUT hmo, LPVOID pReserved)                                          |
| ttui     | MMRESULT    | <i>mididDisconnect</i> (HMIDI hMidi, HMIDIOUT hmo, LPVOID pReserved)                                       |
| ttuiui   | MMRESULT    | <i>midiinAddBuffer</i> (HMIDIIN hMidiIn, LPMIDIHDR lpMidiInHdr, UINT cbMidiInHdr)                          |
| tui      | MMRESULT    | <i>midiinClose</i> (HMIDIIN hMidiIn)                                                                       |
| ttuiui   | MMRESULT    | <i>midiinGetDevCaps</i> (UINT_PTR uDeviceID, LPMIDIINCAPS lpMidiInCaps, UINT cbMidiInCaps)                 |
|          |             |                                                                                                            |

|           |          |                                                                                                                                     |
|-----------|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| ttuiui    | MMRESULT | <a href="#">midiInGetDevCapsA</a> (UINT_PTR uDeviceID, LPMIDIINCAPS lpMidiInCaps, UINT cbMidiInCaps)                                |
| ttuiui    | MMRESULT | <a href="#">midiInGetDevCapsW</a> (UINT_PTR uDeviceID, LPMIDIINCAPS lpMidiInCaps, UINT cbMidiInCaps)                                |
| uisuiui   | MMRESULT | <a href="#">midiInGetErrorText</a> (MMRESULT wError, LPTSTR lpText, UINT cchText)                                                   |
| uiauiui   | MMRESULT | <a href="#">midiInGetErrorTextA</a> (MMRESULT wError, LPSTR lpText, UINT cchText)                                                   |
| uiwuiui   | MMRESULT | <a href="#">midiInGetErrorTextW</a> (MMRESULT wError, LPWSTR lpText, UINT cchText)                                                  |
| ttui      | MMRESULT | <a href="#">midiInGetID</a> (HMIDIIN hmi, LPUINT puDeviceID)                                                                        |
| ui        | UINT     | <a href="#">midiInGetNumDevs</a> (void)                                                                                             |
| tuittui   | DWORD    | <a href="#">midiInMessage</a> (HMIDIIN deviceID, UINT msg, DWORD_PTR dw1, DWORD_PTR dw2)                                            |
| tuittuiui | MMRESULT | <a href="#">midiInOpen</a> (LPHMIDIIN lphMidiIn, UINT uDeviceID, DWORD_PTR dwCallback, DWORD_PTR dwCallbackInstance, DWORD dwFlags) |
| ttuiui    | MMRESULT | <a href="#">midiInPrepareHeader</a> (HMIDIIN hMidiIn, LPMIDIHDR lpMidiInHdr, UINT cbMidiInHdr)                                      |
| tui       | MMRESULT | <a href="#">midiInReset</a> (HMIDIIN hMidiIn)                                                                                       |
| tui       | MMRESULT | <a href="#">midiInStart</a> (HMIDIIN hMidiIn)                                                                                       |
| tui       | MMRESULT | <a href="#">midiInStop</a> (HMIDIIN hMidiIn)                                                                                        |
| ttuiui    | MMRESULT | <a href="#">midiInUnprepareHeader</a> (HMIDIIN hMidiIn, LPMIDIHDR lpMidiInHdr, UINT cbMidiInHdr)                                    |
| tuituiui  | MMRESULT | <a href="#">midiOutCacheDrumPatches</a> (HMIDIOUT hmo, UINT wPatch, WORD *lpKeyArray, UINT wFlags)                                  |
| tuituiui  | MMRESULT | <a href="#">midiOutCachePatches</a> (HMIDIOUT hmo, UINT wBank, WORD *lpPatchArray, UINT wFlags)                                     |
| tui       | MMRESULT | <a href="#">midiOutClose</a> (HMIDIOUT hmo)                                                                                         |
| ttuiui    | MMRESULT | <a href="#">midiOutGetDevCaps</a> (UINT_PTR uDeviceID, LPMIDIOUTCAPS lpMidiOutCaps, UINT cbMidiOutCaps)                             |
| ttuiui    | MMRESULT | <a href="#">midiOutGetDevCapsA</a> (UINT_PTR uDeviceID, LPMIDIOUTCAPS lpMidiOutCaps, UINT cbMidiOutCaps)                            |
| ttuiui    | MMRESULT | <a href="#">midiOutGetDevCapsW</a> (UINT_PTR uDeviceID, LPMIDIOUTCAPS lpMidiOutCaps, UINT cbMidiOutCaps)                            |
| uisuiui   | UINT     | <a href="#">midiOutGetErrorText</a> (MMRESULT mmrError, LPTSTR lpText, UINT cchText)                                                |
| uiauiui   | UINT     | <a href="#">midiOutGetErrorTextA</a> (MMRESULT mmrError, LPSTR lpText, UINT cchText)                                                |
| uiwuiui   | UINT     | <a href="#">midiOutGetErrorTextW</a> (MMRESULT mmrError, LPWSTR lpText, UINT cchText)                                               |

|            |          |                                                                                                                                                   |
|------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| ttui       | MMRESULT | <a href="#">midiOutGetID</a> (HMIDIOUT hmo, LPUINT puDeviceID)                                                                                    |
| ui         | UINT     | <a href="#">midiOutGetNumDevs</a> (void)                                                                                                          |
| ttui       | MMRESULT | <a href="#">midiOutGetVolume</a> (HMIDIOUT hmo, LPDWORD lpdwVolume)                                                                               |
| ttuiui     | MMRESULT | <a href="#">midiOutLongMsg</a> (HMIDIOUT hmo, LPMIDIHDR lpMidiOutHdr, UINT cbMidiOutHdr)                                                          |
| tuittui    | DWORD    | <a href="#">midiOutMessage</a> (HMIDIOUT deviceID, UINT msg, DWORD_PTR dw1, DWORD_PTR dw2)                                                        |
| tuittuiui  | MMRESULT | <a href="#">midiOutOpen</a> (LPHMIDIOUT lphmo, UINT uDeviceID, DWORD_PTR dwCallback, DWORD_PTR dwCallbackInstance, DWORD dwFlags)                 |
| ttuiui     | MMRESULT | <a href="#">midiOutPrepareHeader</a> (HMIDIOUT hmo, LPMIDIHDR lpMidiOutHdr, UINT cbMidiOutHdr)                                                    |
| tui        | MMRESULT | <a href="#">midiOutReset</a> (HMIDIOUT hmo)                                                                                                       |
| tuiui      | MMRESULT | <a href="#">midiOutSetVolume</a> (HMIDIOUT hmo, DWORD dwVolume)                                                                                   |
| tuiui      | MMRESULT | <a href="#">midiOutShortMsg</a> (HMIDIOUT hmo, DWORD dwMsg)                                                                                       |
| ttuiui     | MMRESULT | <a href="#">midiOutUnprepareHeader</a> (HMIDIOUT hmo, LPMIDIHDR lpMidiOutHdr, UINT cbMidiOutHdr)                                                  |
| tui        | MMRESULT | <a href="#">midiStreamClose</a> (HMIDISTRM hStream)                                                                                               |
| ttuittuiui | MMRESULT | <a href="#">midiStreamOpen</a> (LPHMIDISTRM lphStream, LPUINT puDeviceID, DWORD cMidi, DWORD_PTR dwCallback, DWORD_PTR dwInstance, DWORD fdwOpen) |
| ttuiui     | MMRESULT | <a href="#">midiStreamOut</a> (HMIDISTRM hMidiStream, LPMIDIHDR lpMidiHdr, UINT cbMidiHdr)                                                        |
| tui        | MMRESULT | <a href="#">midiStreamPause</a> (HMIDISTRM hms)                                                                                                   |
| ttuiui     | MMRESULT | <a href="#">midiStreamPosition</a> (HMIDISTRM hms, LPMMTIME pmmt, UINT cbmmt)                                                                     |
| ttuiui     | MMRESULT | <a href="#">midiStreamProperty</a> (HMIDISTRM hm, LPBYTE lppropdata, DWORD dwProperty)                                                            |
| tui        | MMRESULT | <a href="#">midiStreamRestart</a> (HMIDISTRM hms)                                                                                                 |
| tui        | MMRESULT | <a href="#">midiStreamStop</a> (HMIDISTRM hms)                                                                                                    |
| tui        | MMRESULT | <a href="#">mixerClose</a> (HMIXER hmx)                                                                                                           |
| ttuiui     | MMRESULT | <a href="#">mixerGetControlDetails</a> (HMIXEROBJ hmxobj, LPMIXERCONTROLDETAILS pmxcd, DWORD fdwDetails)                                          |
| ttuiui     | MMRESULT | <a href="#">mixerGetControlDetailsA</a> (HMIXEROBJ hmxobj, LPMIXERCONTROLDETAILS pmxcd, DWORD fdwDetails)                                         |
| ttuiui     | MMRESULT | <a href="#">mixerGetControlDetailsW</a> (HMIXEROBJ hmxobj, LPMIXERCONTROLDETAILS pmxcd, DWORD fdwDetails)                                         |
| ttuiui     | MMRESULT | <a href="#">mixerGetDevCaps</a> (UINT_PTR uMxId, LPMIXERCAPS pmxcaps, UINT cbmxcaps)                                                              |
|            |          |                                                                                                                                                   |

|           |            |                                                                                                                  |
|-----------|------------|------------------------------------------------------------------------------------------------------------------|
| ttuiui    | MMRESULT   | <a href="#">mixerGetDevCapsA</a> (UINT_PTR uMxId, LPMIXERCAPS pmxcaps, UINT cbmxcaps)                            |
| ttuiui    | MMRESULT   | <a href="#">mixerGetDevCapsW</a> (UINT_PTR uMxId, LPMIXERCAPS pmxcaps, UINT cbmxcaps)                            |
| ttuiui    | MMRESULT   | <a href="#">mixerGetID</a> (HMIXEROBJ hmxobj, UINT FAR *puMxId, DWORD fdwId)                                     |
| ttuiui    | MMRESULT   | <a href="#">mixerGetLineControl</a> (HMIXEROBJ hmxobj, LPMIXERLINECONTROLS pmxlc, DWORD fdwControls)             |
| ttuiui    | MMRESULT   | <a href="#">mixerGetLineControlsA</a> (HMIXEROBJ hmxobj, LPMIXERLINECONTROLS pmxlc, DWORD fdwControls)           |
| ttuiui    | MMRESULT   | <a href="#">mixerGetLineControlsW</a> (HMIXEROBJ hmxobj, LPMIXERLINECONTROLS pmxlc, DWORD fdwControls)           |
| ttuiui    | MMRESULT   | <a href="#">mixerGetLineInfo</a> (HMIXEROBJ hmxobj, LPMIXERLINE pmxl, DWORD fdwInfo)                             |
| ttuiui    | MMRESULT   | <a href="#">mixerGetLineInfoA</a> (HMIXEROBJ hmxobj, LPMIXERLINE pmxl, DWORD fdwInfo)                            |
| ttuiui    | MMRESULT   | <a href="#">mixerGetLineInfoW</a> (HMIXEROBJ hmxobj, LPMIXERLINE pmxl, DWORD fdwInfo)                            |
| ui        | UINT       | <a href="#">mixerGetNumDevs</a> (void)                                                                           |
| tuittui   | DWORD      | <a href="#">mixerMessage</a> (HMIXER driverID, UINT uMsg, DWORD_PTR dwParam1, DWORD_PTR dwParam2)                |
| tuittuiui | MMRESULT   | <a href="#">mixerOpen</a> (LPHMIXER phmx, UINT uMxId, DWORD_PTR dwCallback, DWORD_PTR dwInstance, DWORD fdwOpen) |
| ttuiui    | MMRESULT   | <a href="#">mixerSetControlDetails</a> (HMIXEROBJ hmxobj, LPMIXERCONTROLDETAILS pmxcd, DWORD fdwDetails)         |
| ui        | DWORD      | <a href="#">mmGetCurrentTask</a> (void)                                                                          |
| ttuiui    | MMRESULT   | <a href="#">mmioAdvance</a> (HMMIO hmmio, LPMMIOINFO lpmmioinfo, UINT wFlags)                                    |
| ttuiui    | MMRESULT   | <a href="#">mmioAscend</a> (HMMIO hmmio, LPMMCKINFO lpck, UINT wFlags)                                           |
| tuiui     | MMRESULT   | <a href="#">mmioClose</a> (HMMIO hmmio, UINT wFlags)                                                             |
| ttuiui    | MMRESULT   | <a href="#">mmioCreateChunk</a> (HMMIO hmmio, LPMMCKINFO lpck, UINT wFlags)                                      |
| ttuiui    | MMRESULT   | <a href="#">mmioDescend</a> (HMMIO hmmio, LPMMCKINFO lpck, LPMMCKINFO lpckParent, UINT wFlags)                   |
| tuiui     | MMRESULT   | <a href="#">mmioFlush</a> (HMMIO hmmio, UINT fuFlush)                                                            |
| ttuiui    | MMRESULT   | <a href="#">mmioGetInfo</a> (HMMIO hmmio, LPMMIOINFO lpmmioinfo, UINT wFlags)                                    |
| uituit    | LPMMIOPROC | <a href="#">mmioInstallIOProc</a> (FOURCC fccIOProc, LPMMIOPROC piOProc, DWORD dwFlags)                          |
| uituit    | LPMMIOPROC | <a href="#">mmioInstallIOProcA</a> (FOURCC fccIOProc, LPMMIOPROC piOProc, DWORD dwFlags)                         |

|          |            |                                                                                                                        |
|----------|------------|------------------------------------------------------------------------------------------------------------------------|
| uituit   | LPMMIOPROC | <code>mmioInstallIOProcW</code> (FOURCC fccIOProc, LPMMIOPROC pIOProc, DWORD dwFlags)                                  |
| stuit    | HMMIO      | <code>mmioOpen</code> (LPTSTR szFilename, LPMMIOINFO lpmmioinfo, DWORD dwOpenFlags)                                    |
| atuit    | HMMIO      | <code>mmioOpenA</code> (LPSTR szFilename, LPMMIOINFO lpmmioinfo, DWORD dwOpenFlags)                                    |
| wtuit    | HMMIO      | <code>mmioOpenW</code> (LPWSTR szFilename, LPMMIOINFO lpmmioinfo, DWORD dwOpenFlags)                                   |
| ttuiui   | LONG       | <code>mmioReac</code> (HMMIO hmmio, HPSTR pch, LONG cch)                                                               |
| sstuiui  | MMRESULT   | <code>mmioRename</code> (LPCTSTR szFilename, LPCTSTR szNewFilename, const LPMMIOINFO lpmmioinfo, DWORD dwRenameFlags)  |
| aatuiui  | MMRESULT   | <code>mmioRenameA</code> (LPCSTR szFilename, LPCSTR szNewFilename, const LPMMIOINFO lpmmioinfo, DWORD dwRenameFlags)   |
| wwtuiui  | MMRESULT   | <code>mmioRenameW</code> (LPCWSTR szFilename, LPCWSTR szNewFilename, const LPMMIOINFO lpmmioinfo, DWORD dwRenameFlags) |
| tuiiui   | LONG       | <code>mmioSeek</code> (HMMIO hmmio, LONG lOffset, int iOrigin)                                                         |
| tuittt   | LRESULT    | <code>mmioSendMessage</code> (HMMIO hmmio, UINT wParam, LPARAM lParam1, LPARAM lParam2)                                |
| tauiuiui | MMRESULT   | <code>mmioSetBuffer</code> (HMMIO hmmio, LPSTR pchBuffer, LONG cchBuffer, UINT wFlags)                                 |
| ttuiui   | MMRESULT   | <code>mmioSetInfo</code> (HMMIO hmmio, LPMMIOINFO lpmmioinfo, UINT wFlags)                                             |
| suiui    | FOURCC     | <code>mmioStringToFOURCC</code> (LPCTSTR sz, UINT wFlags)                                                              |
| aiuiui   | FOURCC     | <code>mmioStringToFOURCCA</code> (LPCSTR sz, UINT wFlags)                                                              |
| wuiui    | FOURCC     | <code>mmioStringToFOURCCW</code> (LPCWSTR sz, UINT wFlags)                                                             |
| ttuiui   | LONG       | <code>mmioWrite</code> (HMMIO hmmio, char _huge *pch, LONG cch)                                                        |
| ui       | UINT       | <code>mmSystemGetVersion</code> (void)                                                                                 |
| uii      | VOID       | <code>mmTaskBlock</code> (DWORD h)                                                                                     |
| ttui     | UINT       | <code>mmTaskCreate</code> (LPTASKCALLBACK lpfn, HANDLE FAR *lph, DWORD_PTR dwInst)                                     |
| i        | VOID       | <code>mmTaskYield</code> (void)                                                                                        |
| wwtt     | HDRVR      | <code>OpenDriver</code> (_In_ LPCWSTR lpDriverName, _In_ LPCWSTR lpSectionName, _In_ LPARAM lParam)                    |
| stuii    | BOOL       | <code>PlaySound</code> (LPCTSTR pszSound, HMODULE hmod, DWORD fdwSound)                                                |
| atuii    | BOOL       | <code>PlaySoundA</code> (LPCSTR pszSound, HMODULE hmod, DWORD fdwSound)                                                |
| wtuii    | BOOL       | <code>PlaySoundW</code> (LPCWSTR pszSound, HMODULE hmod, DWORD fdwSound)                                               |

|           |          |                                                                                                                                    |
|-----------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| tuittt    | LRESULT  | SendDriverMessage(_In_ HDRVR hdrvr, _In_ UINT msg, _Inout_ LPARAM lParam1, _Inout_ LPARAM lParam2)                                 |
| suii      | BOOL     | sndPlaySound(LPCTSTR lpszSound, UINT fuSound)                                                                                      |
| auui      | BOOL     | sndPlaySoundA(LPCSTR lpszSound, UINT fuSound)                                                                                      |
| wuii      | BOOL     | sndPlaySoundW(LPCWSTR lpszSound, UINT fuSound)                                                                                     |
| uiui      | MMRESULT | timeBeginPeriod(UINT uPeriod)                                                                                                      |
| uiui      | MMRESULT | timeEndPeriod(UINT uPeriod)                                                                                                        |
| tuiui     | MMRESULT | timeGetDevCaps(LPTIMECAPS ptc, UINT cbtc)                                                                                          |
| tuiui     | MMRESULT | timeGetSystemTime(LPMMTIME pmmt, UINT cbmmt)                                                                                       |
| ui        | DWORD    | timeGetTime(void)                                                                                                                  |
| uiui      | MMRESULT | timeKillEvent(UINT uTimerID)                                                                                                       |
| uiuittuiu | MMRESULT | timeSetEvent(UINT uDelay, UINT uResolution, LPTIMECALLBACK lpTimeProc, DWORD_PTR dwUser, UINT fuEvent)                             |
| ttuiui    | MMRESULT | waveInAddBuffer(HWAVEIN hwi, LPWAVEHDR pwh, UINT cbwh)                                                                             |
| tui       | MMRESULT | waveInClose(HWAVEIN hwi)                                                                                                           |
| ttuiui    | MMRESULT | waveInGetDevCaps(UINT_PTR uDeviceID, LPWAVEINCAPS pwic, UINT cbwic)                                                                |
| ttuiui    | MMRESULT | waveInGetDevCapsA(UINT_PTR uDeviceID, LPWAVEINCAPS pwic, UINT cbwic)                                                               |
| ttuiui    | MMRESULT | waveInGetDevCapsW(UINT_PTR uDeviceID, LPWAVEINCAPS pwic, UINT cbwic)                                                               |
| uisuiui   | MMRESULT | waveInGetErrorText(MMRESULT mmrError, LPTSTR pszText, UINT cchText)                                                                |
| uiauiui   | MMRESULT | waveInGetErrorTextA(MMRESULT mmrError, LPSTR pszText, UINT cchText)                                                                |
| uiwuiui   | MMRESULT | waveInGetErrorTextW(MMRESULT mmrError, LPWSTR pszText, UINT cchText)                                                               |
| ttui      | MMRESULT | waveInGetID(HWAVEIN hwi, LPUINT puDeviceID)                                                                                        |
| ui        | UINT     | waveInGetNumDevs(void)                                                                                                             |
| ttuiui    | MMRESULT | waveInGetPosition(HWAVEIN hwi, LPMMTIME pmmt, UINT cbmmt)                                                                          |
| tuittui   | DWORD    | waveInMessage(HWAVEIN deviceID, UINT uMsg, DWORD_PTR dwParam1, DWORD_PTR dwParam2)                                                 |
| tuitttuiu | MMRESULT | waveInOpen(LPHWAVEIN phwi, UINT uDeviceID, LPCWAVEFORMATEX pwf, DWORD_PTR dwCallback, DWORD_PTR dwCallbackInstance, DWORD fdwOpen) |
| ttuiui    | MMRESULT | waveInPrepareHeader(HWAVEIN hwi, LPWAVEHDR pwh, UINT cbwh)                                                                         |

|          |          |                                                                                                                                         |
|----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| tui      | MMRESULT | waveInReset(HWAVEIN hwi)                                                                                                                |
| tui      | MMRESULT | waveInStart(HWAVEIN hwi)                                                                                                                |
| tui      | MMRESULT | waveInStop(HWAVEIN hwi)                                                                                                                 |
| ttuiui   | MMRESULT | waveInUnprepareHeader(HWAVEIN hwi, LPWAVEHDR pwh, UINT cbwh)                                                                            |
| tui      | MMRESULT | waveOutBreakLoop(HWAVEOUT hwo)                                                                                                          |
| tui      | MMRESULT | waveOutClose(HWAVEOUT hwo)                                                                                                              |
| ttuiui   | MMRESULT | waveOutGetDevCaps(UINT_PTR uDeviceID, LPWAVEOUTCAPS pwoc, UINT cbwoc)                                                                   |
| ttuiui   | MMRESULT | waveOutGetDevCapsA(UINT_PTR uDeviceID, LPWAVEOUTCAPS pwoc, UINT cbwoc)                                                                  |
| ttuiui   | MMRESULT | waveOutGetDevCapsW(UINT_PTR uDeviceID, LPWAVEOUTCAPS pwoc, UINT cbwoc)                                                                  |
| uisuiui  | MMRESULT | waveOutGetErrorText(MMRESULT mmrError, LPTSTR pszText, UINT cchText)                                                                    |
| uiauiui  | MMRESULT | waveOutGetErrorTextA(MMRESULT mmrError, LPSTR pszText, UINT cchText)                                                                    |
| uiwuiui  | MMRESULT | waveOutGetErrorTextW(MMRESULT mmrError, LPWSTR pszText, UINT cchText)                                                                   |
| ttui     | MMRESULT | waveOutGetID(HWAVEOUT hwo, LPUINT puDeviceID)                                                                                           |
| ui       | UINT     | waveOutGetNumDevs(void)                                                                                                                 |
| ttui     | MMRESULT | waveOutGetPitch(HWAVEOUT hwo, LPDWORD pdwPitch)                                                                                         |
| ttui     | MMRESULT | waveOutGetPlaybackRate(HWAVEOUT hwo, LPDWORD pdwRate)                                                                                   |
| ttuiui   | MMRESULT | waveOutGetPosition(HWAVEOUT hwo, LPMMTIME pmmt, UINT cbmmt)                                                                             |
| ttui     | MMRESULT | waveOutGetVolume(HWAVEOUT hwo, LPDWORD pdwVolume)                                                                                       |
| tuittui  | DWORD    | waveOutMessage(HWAVEOUT deviceID, UINT uMsg, DWORD_PTR dwParam1, DWORD_PTR dwParam2)                                                    |
| ttttuiui | MMRESULT | waveOutOpen(LPHWAVEOUT phwo, UINT_PTR uDeviceID, LPWAVEFORMATEX pwx, DWORD_PTR dwCallback, DWORD_PTR dwCallbackInstance, DWORD fdwOpen) |
| tui      | MMRESULT | waveOutPause(HWAVEOUT hwo)                                                                                                              |
| ttuiui   | MMRESULT | waveOutPrepareHeader(HWAVEOUT hwo, LPWAVEHDR pwh, UINT cbwh)                                                                            |
| tui      | MMRESULT | waveOutReset(HWAVEOUT hwo)                                                                                                              |
| tui      | MMRESULT | waveOutRestart(HWAVEOUT hwo)                                                                                                            |
| tuiui    | MMRESULT | waveOutSetPitch(HWAVEOUT hwo, DWORD dwPitch)                                                                                            |
| tuiui    | MMRESULT | waveOutSetPlaybackRate(HWAVEOUT hwo, DWORD dwRate)                                                                                      |
|          |          |                                                                                                                                         |

|        |          |                                                                                 |
|--------|----------|---------------------------------------------------------------------------------|
| tuiui  | MMRESULT | <a href="#">waveOutSetVolume</a> (HWAVEOUT hwo, DWORD dwVolume)                 |
| ttuiui | MMRESULT | <a href="#">waveOutUnprepareHeader</a> (HWAVEOUT hwo, LPWAVEHDR pwh, UINT cbwh) |
| ttuiui | MMRESULT | <a href="#">waveOutWrite</a> (HWAVEOUT hwo, LPWAVEHDR pwh, UINT cbwh)           |

## Ws2\_32.dll

|              |          |                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tii          | int      | __WSAFDLSOCKET (SOCKET fd, fd_set *set)                                                                                                                                                                                                                                                                                                                          |
| tttt         | SOCKET   | accept(_In_ SOCKET s, _Out_ struct sockaddr *addr, _Inout_ int *addrlen)                                                                                                                                                                                                                                                                                         |
| ttii         | int      | bind(_In_ SOCKET s, _In_ const struct sockaddr *name, _In_ int namelen)                                                                                                                                                                                                                                                                                          |
| ti           | int      | closesocket(_In_ SOCKET s)                                                                                                                                                                                                                                                                                                                                       |
| ttii         | int      | connect(_In_ SOCKET s, _In_ const struct sockaddr *name, _In_ int namelen)                                                                                                                                                                                                                                                                                       |
| ti           | VOID     | freeaddrinfo(_In_ struct addrinfo *ai)                                                                                                                                                                                                                                                                                                                           |
| ti           | VOID     | FreeAddrInfoEx(_In_ PADDRINFOEX pAddrInfo)                                                                                                                                                                                                                                                                                                                       |
| ti           | VOID     | FreeAddrInfoExW(_In_ PADDRINFOEX pAddrInfo)                                                                                                                                                                                                                                                                                                                      |
| ti           | VOID     | FreeAddrInfoW(_In_ PADDRINFOW pAddrInfo)                                                                                                                                                                                                                                                                                                                         |
| aatti        | int      | getaddrinfo(_In_opt_ PCSTR pNodeName, _In_opt_ PCSTR pServiceName, _In_opt_ const ADDRINFOA *pHints, _Out_ PADDRINFOA *ppResult)                                                                                                                                                                                                                                 |
| ssuitttttti  | int      | GetAddrInfoEx(_In_opt_ PCTSTR pNodeName, _In_opt_ PCTSTR pServiceName, _In_ DWORD dwNameSpace, _In_opt_ LPGUID lpNspId, _In_opt_ const ADDRINFOEX *pHints, _Out_ PADDRINFOEX *ppResult, _In_opt_ struct timeval *timeout, _In_opt_ LPOVERLAPPED lpOverlapped, _In_opt_ LPLOOKUPSERVICE_COMPLETION_ROUTINE lpCompletionRoutine, _Out_opt_ LPHANDLE lpNameHandle)  |
| aauiitttttti | int      | GetAddrInfoExA(_In_opt_ PCTSTR pNodeName, _In_opt_ PCTSTR pServiceName, _In_ DWORD dwNameSpace, _In_opt_ LPGUID lpNspId, _In_opt_ const ADDRINFOEX *pHints, _Out_ PADDRINFOEX *ppResult, _In_opt_ struct timeval *timeout, _In_opt_ LPOVERLAPPED lpOverlapped, _In_opt_ LPLOOKUPSERVICE_COMPLETION_ROUTINE lpCompletionRoutine, _Out_opt_ LPHANDLE lpNameHandle) |
| wwuitttttti  | int      | GetAddrInfoExW(_In_opt_ PCTSTR pNodeName, _In_opt_ PCTSTR pServiceName, _In_ DWORD dwNameSpace, _In_opt_ LPGUID lpNspId, _In_opt_ const ADDRINFOEX *pHints, _Out_ PADDRINFOEX *ppResult, _In_opt_ struct timeval *timeout, _In_opt_ LPOVERLAPPED lpOverlapped, _In_opt_ LPLOOKUPSERVICE_COMPLETION_ROUTINE lpCompletionRoutine, _Out_opt_ LPHANDLE lpNameHandle) |
| wwtti        | int      | GetAddrInfoW(_In_opt_ PCWSTR pNodeName, _In_opt_ PCWSTR pServiceName, _In_opt_ const ADDRINFOW *pHints, _Out_ PADDRINFOW *ppResult)                                                                                                                                                                                                                              |
| aiit         | hostent* | gethostbyaddr(_In_ const char *addr, _In_ int len, _In_ int type)                                                                                                                                                                                                                                                                                                |

|            |              |                                                                                                                                                                                                             |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| at         | hostent*     | gethostbyname(_In_ const char *name)                                                                                                                                                                        |
| aii        | int          | gethostname(_Out_ char *name, _In_ int namelen)                                                                                                                                                             |
| tiauiiuiii | int          | getnameinfo(_In_ const struct sockaddr FAR *sa, _In_ socklen_t salen, _Out_ char FAR *host, _In_ DWORD hostlen, _Out_ char FAR *serv, _In_ DWORD servlen, _In_ int flags)                                   |
| tiwuiwuiii | int          | GetNameInfoW(_In_ const SOCKADDR *pSockaddr, _In_ socklen_t SockaddrLength, _Out_ PWCHAR pNodeBuffer, _In_ DWORD NodeBufferSize, _Out_ PWCHAR pServiceBuffer, _In_ DWORD ServiceBufferSize, _In_ INT Flags) |
| titi       | int          | getpeername(_In_ SOCKET s, _Out_ struct sockaddr *name, _Inout_ int *namelen)                                                                                                                               |
| at         | PROTOENT*    | getprotobyname(_In_ const char *name)                                                                                                                                                                       |
| it         | PROTOENT*    | getprotobynumber(_In_ int number)                                                                                                                                                                           |
| aat        | servent*     | getservbyname(_In_ const char *name, _In_ const char *proto)                                                                                                                                                |
| iat        | servent*     | getservbyport(_In_ int port, _In_ const char *proto)                                                                                                                                                        |
| titi       | int          | getsockname(_In_ SOCKET s, _Out_ struct sockaddr *name, _Inout_ int *namelen)                                                                                                                               |
| tiiati     | int          | getsockopt(_In_ SOCKET s, _In_ int level, _In_ int optname, _Out_ char *optval, _Inout_ int *optlen)                                                                                                        |
| uiui       | u_long       | htonl(_In_ u_long hostlong)                                                                                                                                                                                 |
| uhuh       | u_short      | htons(_In_ u_short hostshort)                                                                                                                                                                               |
| aii        | unsignedlong | inet_addr(_In_ const char *cp)                                                                                                                                                                              |
| uit        | char*        | inet_ntoa(_In_ struct in_addr in)                                                                                                                                                                           |
| itsts      | PCTSTR       | inet_ntop(_In_ INT Family, _In_ PVOID pAddr, _Out_ PTSTR pStringBuf, _In_ size_t StringBufSize)                                                                                                             |
| itwtw      | PCTSTR       | inet_ntopW(_In_ INT Family, _In_ PVOID pAddr, _Out_ PTSTR pStringBuf, _In_ size_t StringBufSize)                                                                                                            |
| isti       | int          | inet_pton(_In_ INT Family, _In_ PCTSTR pszAddrString, _Out_ PVOID pAddrBuf)                                                                                                                                 |
| iwti       | int          | inet_ptonW(_In_ INT Family, _In_ PCTSTR pszAddrString, _Out_ PVOID pAddrBuf)                                                                                                                                |
| tuiti      | int          | ioctlsocket(_In_ SOCKET s, _In_ long cmd, _Inout_ u_long *argp)                                                                                                                                             |
| tii        | int          | listen(_In_ SOCKET s, _In_ int backlog)                                                                                                                                                                     |
| uiui       | u_long       | ntohl(_In_ u_long netlong)                                                                                                                                                                                  |
| uhuh       | u_short      | ntohs(_In_ u_short netshort)                                                                                                                                                                                |
| taiii      | int          | recv(_In_ SOCKET s, _Out_ char *buf, _In_ int len, _In_ int flags)                                                                                                                                          |
| taiitti    | int          | recvfrom(_In_ SOCKET s, _Out_ char *buf, _In_ int len, _In_ int flags, _Out_ struct sockaddr *from, _Inout_opt_ int *fromlen)                                                                               |
|            |              | select(_In_ int nfds, _Inout_ fd_set *readfds, _Inout_ fd_set                                                                                                                                               |

|                 |        |                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| itttti          | int    | *writefds, _Inout_ fd_set *exceptfds, _In_ const struct timeval *timeout)                                                                                                                                                                                                                                                                                                                       |
| taiii           | int    | send(_In_ SOCKET s, _In_ const char *buf, _In_ int len, _In_ int flags)                                                                                                                                                                                                                                                                                                                         |
| taiitii         | int    | sendto(_In_ SOCKET s, _In_ const char *buf, _In_ int len, _In_ int flags, _In_ const struct sockaddr *to, _In_ int tolen)                                                                                                                                                                                                                                                                       |
| sstuituiittttti | int    | SetAddrInfoEx(_In_ PCTSTR pName, _In_ PCTSTR pServiceName, _Inout_ SOCKET_ADDRESS *pAddresses, _In_ DWORD dwAddressCount, _In_opt_ LPBLOB lpBlob, _In_ DWORD dwFlags, _In_ DWORD dwNameSpace, _In_opt_ LPGUID lpNspId, _In_opt_ struct timeval *timeout, _In_opt_ LPOVERLAPPED lpOverlapped, _In_opt_ LPLOOKUPSERVICE_COMPLETION_ROUTINE lpCompletionRoutine, _Out_opt_ LPHANDLE lpNameHandle)  |
| aatuituiittttti | int    | SetAddrInfoExA(_In_ PCTSTR pName, _In_ PCTSTR pServiceName, _Inout_ SOCKET_ADDRESS *pAddresses, _In_ DWORD dwAddressCount, _In_opt_ LPBLOB lpBlob, _In_ DWORD dwFlags, _In_ DWORD dwNameSpace, _In_opt_ LPGUID lpNspId, _In_opt_ struct timeval *timeout, _In_opt_ LPOVERLAPPED lpOverlapped, _In_opt_ LPLOOKUPSERVICE_COMPLETION_ROUTINE lpCompletionRoutine, _Out_opt_ LPHANDLE lpNameHandle) |
| wwtuituiittttti | int    | SetAddrInfoExW(_In_ PCTSTR pName, _In_ PCTSTR pServiceName, _Inout_ SOCKET_ADDRESS *pAddresses, _In_ DWORD dwAddressCount, _In_opt_ LPBLOB lpBlob, _In_ DWORD dwFlags, _In_ DWORD dwNameSpace, _In_opt_ LPGUID lpNspId, _In_opt_ struct timeval *timeout, _In_opt_ LPOVERLAPPED lpOverlapped, _In_opt_ LPLOOKUPSERVICE_COMPLETION_ROUTINE lpCompletionRoutine, _Out_opt_ LPHANDLE lpNameHandle) |
| tiaaii          | int    | setsockopt(_In_ SOCKET s, _In_ int level, _In_ int optname, _In_ const char *optval, _In_ int optlen)                                                                                                                                                                                                                                                                                           |
| tii             | int    | shutdown(_In_ SOCKET s, _In_ int how)                                                                                                                                                                                                                                                                                                                                                           |
| iiit            | SOCKET | socket(_In_ int af, _In_ int type, _In_ int protocol)                                                                                                                                                                                                                                                                                                                                           |
| ttttt           | SOCKET | WSAAccept(_In_ SOCKET s, _Out_ struct sockaddr *addr, _Inout_ LPINT addrlen, _In_ LPCONDITIONPROC lpfnCondition, _In_ DWORD_PTR dwCallbackData)                                                                                                                                                                                                                                                 |
| tuitsti         | int    | WSAAddressToString(_In_ LPSOCKADDR lpsaAddress, _In_ DWORD dwAddressLength, _In_opt_ LPWSAPROTOCOL_INFO lpProtocolInfo, _Inout_ LPCTSTR lpszAddressString, _Inout_ LPDWORD lpdwAddressStringLength)                                                                                                                                                                                             |
| tuitati         | int    | WSAAddressToStringA(_In_ LPSOCKADDR lpsaAddress, _In_ DWORD dwAddressLength, _In_opt_ LPWSAPROTOCOL_INFO lpProtocolInfo, _Inout_ LPSTR lpszAddressString, _Inout_ LPDWORD lpdwAddressStringLength)                                                                                                                                                                                              |
|                 |        | WSAAddressToStringW(_In_ LPSOCKADDR lpsaAddress, _In_                                                                                                                                                                                                                                                                                                                                           |

|           |        |                                                                                                                                                                                                                                                                                         |
|-----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuitwti   | int    | DWORD dwAddressLength, _In_opt_ LPWSAPROTOCOL_INFO lpProtocolInfo, _Inout_ LPWSTR lpszAddressString, _Inout_ LPDWORD lpdwAddressStringLength)                                                                                                                                           |
| titi      | int    | WSAAdvertiseProvider(_In_ const GUID *puuidProviderId, _In_ const LPCNSPV2_ROUTINE *pNSPv2Routine)                                                                                                                                                                                      |
| tuiiaait  | HANDLE | WSAAsyncGetHostByAddr(_In_ HWND hWnd, _In_ unsigned int wMsg, _In_ const char *addr, _In_ int len, _In_ int type, _Out_ char *buf, _In_ int buflen)                                                                                                                                     |
| tuiaait   | HANDLE | WSAAsyncGetHostByName(_In_ HWND hWnd, _In_ unsigned int wMsg, _In_ const char *name, _Out_ char *buf, _In_ int buflen)                                                                                                                                                                  |
| tuiaait   | HANDLE | WSAAsyncGetProtoByName(_In_ HWND hWnd, _In_ unsigned int wMsg, _In_ const char *name, _Out_ char *buf, _Out_ int buflen)                                                                                                                                                                |
| tuiiait   | HANDLE | WSAAsyncGetProtoByNumber(_In_ HWND hWnd, _In_ unsigned int wMsg, _In_ int number, _Out_ char *buf, _In_ int buflen)                                                                                                                                                                     |
| tuiaaaait | HANDLE | WSAAsyncGetServByName(_In_ HWND hWnd, _In_ unsigned int wMsg, _In_ const char *name, _In_ const char *proto, _Out_ char *buf, _In_ int buflen)                                                                                                                                          |
| tuiiaait  | HANDLE | WSAAsyncGetServByPort(_In_ HWND hWnd, _In_ unsigned int wMsg, _In_ int port, _In_ const char *proto, _Out_ char *buf, _In_ int buflen)                                                                                                                                                  |
| ttuiiii   | int    | WSAAsyncSelect(_In_ SOCKET s, _In_ HWND hWnd, _In_ unsigned int wMsg, _In_ long lEvent)                                                                                                                                                                                                 |
| ti        | int    | WSACancelAsyncRequest(_In_ HANDLE hAsyncTaskHandle)                                                                                                                                                                                                                                     |
| i         | int    | WSACleanup(void)                                                                                                                                                                                                                                                                        |
| ti        | BOOL   | WSACloseEvent(_In_ WSAEVENT hEvent)                                                                                                                                                                                                                                                     |
| ttitttti  | int    | WSAConnect(_In_ SOCKET s, _In_ const struct sockaddr *name, _In_ int namelen, _In_ LPWSABUF lpCallerData, _Out_ LPWSABUF lpCalleeData, _In_ LPQOS lpSQOS, _In_ LPQOS lpGQOS)                                                                                                            |
| ttttttti  | BOOL   | WSAConnectByList(_In_ SOCKET s, _In_ PSOCKET_ADDRESS_LIST SocketAddressList, _Inout_ LPDWORD LocalAddressLength, _Out_ LPSOCKADDR LocalAddress, _Inout_ LPDWORD RemoteAddressLength, _Out_ LPSOCKADDR RemoteAddress, _In_ const struct timeval *timeout, _In_ LPWSAOVERLAPPED Reserved) |
| tssttttti | BOOL   | WSAConnectByName(_In_ SOCKET s, _In_ LPTSTR nodename, _In_ LPTSTR servicename, _Inout_ LPDWORD LocalAddressLength, _Out_ LPSOCKADDR LocalAddress, _Inout_ LPDWORD RemoteAddressLength, _Out_ LPSOCKADDR RemoteAddress, _In_ const struct timeval *timeout, LPWSAOVERLAPPED Reserved)    |
| taattttti | BOOL   | WSAConnectByNameA(_In_ SOCKET s, _In_ LPSTR nodename, _In_ LPSTR servicename, _Inout_ LPDWORD LocalAddressLength, _Out_ LPSOCKADDR LocalAddress, _Inout_ LPDWORD RemoteAddressLength, _Out_ LPSOCKADDR RemoteAddress, _In_                                                              |

|           |          |                                                                                                                                                                                                                                                                                         |
|-----------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |          | const struct timeval *timeout, LPWSAOVERLAPPED Reserved)                                                                                                                                                                                                                                |
| twwttttti | BOOL     | WSAConnectByNameW(_In_ SOCKET s, _In_ LPWSTR nodename, _In_ LPWSTR servicename, _Inout_ LPDWORD LocalAddressLength, _Out_ LP SOCKADDR LocalAddress, _Inout_ LPDWORD RemoteAddressLength, _Out_ LP SOCKADDR RemoteAddress, _In_ const struct timeval *timeout, LPWSAOVERLAPPED Reserved) |
| t         | WSAEVENT | WSACreateEvent(void)                                                                                                                                                                                                                                                                    |
| tuiti     | int      | WSADuplicateSocket(_In_ SOCKET s, _In_ DWORD dwProcessId, _Out_ LPWSAPROTOCOL_INFO lpProtocolInfo)                                                                                                                                                                                      |
| tuiti     | int      | WSADuplicateSocketA(_In_ SOCKET s, _In_ DWORD dwProcessId, _Out_ LPWSAPROTOCOL_INFO lpProtocolInfo)                                                                                                                                                                                     |
| tuiti     | int      | WSADuplicateSocketW(_In_ SOCKET s, _In_ DWORD dwProcessId, _Out_ LPWSAPROTOCOL_INFO lpProtocolInfo)                                                                                                                                                                                     |
| titi      | int      | WSAEnumNameSpaceProviders(_Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSANAMESPACE_INFO lpnspBuffer)                                                                                                                                                                                      |
| titi      | int      | WSAEnumNameSpaceProvidersA(_Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSANAMESPACE_INFO lpnspBuffer)                                                                                                                                                                                     |
| titi      | int      | WSAEnumNameSpaceProvidersEx(_Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSANAMESPACE_INFOEX lpnspBuffer)                                                                                                                                                                                  |
| titi      | int      | WSAEnumNameSpaceProvidersExA(_Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSANAMESPACE_INFOEX lpnspBuffer)                                                                                                                                                                                 |
| titi      | int      | WSAEnumNameSpaceProvidersExW(_Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSANAMESPACE_INFOEX lpnspBuffer)                                                                                                                                                                                 |
| titi      | int      | WSAEnumNameSpaceProvidersW(_Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSANAMESPACE_INFO lpnspBuffer)                                                                                                                                                                                     |
| titi      | int      | WSAEnumNetworkEvents(_In_ SOCKET s, _In_ WSAEVENT hEventObject, _Out_ LPWSANETWORKEVENTS lpNetworkEvents)                                                                                                                                                                               |
| titi      | int      | WSAEnumProtocols(_In_ LPINT lpiProtocols, _Out_ LPWSAPROTOCOL_INFO lpProtocolBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                                                                                                                                  |
| titi      | int      | WSAEnumProtocolsA(_In_ LPINT lpiProtocols, _Out_ LPWSAPROTOCOL_INFO lpProtocolBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                                                                                                                                 |
| titi      | int      | WSAEnumProtocolsW(_In_ LPINT lpiProtocols, _Out_ LPWSAPROTOCOL_INFO lpProtocolBuffer, _Inout_ LPDWORD lpdwBufferLength)                                                                                                                                                                 |
| ttuii     | int      | WSAEventSelect(_In_ SOCKET s, _In_ WSAEVENT hEventObject, _In_ long lNetworkEvents)                                                                                                                                                                                                     |
|           |          |                                                                                                                                                                                                                                                                                         |

|          |        |                                                                                                                                                                                                                                                                                         |
|----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| i        | int    | WSAGetLastError(void)                                                                                                                                                                                                                                                                   |
| ttiti    | BOOL   | WSAGetOverlappedResult(_In_ SOCKET s, _In_ LPWSAOVERLAPPED lpOverlapped, _Out_ LPDWORD lpcbTransfer, _In_ BOOL fWait, _Out_ LPDWORD lpdwFlags)                                                                                                                                          |
| titi     | BOOL   | WSAGetQOSByName(_In_ SOCKET s, _Inout_ LPWSABUF lpQOSName, _Out_ LPQOS lpQOS)                                                                                                                                                                                                           |
| ttti     | int    | WSAGetServiceClassInfo(_In_ LPGUID lpProviderId, _In_ LPGUID lpServiceClassId, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSASERVICECLASSINFO lpServiceClassInfo)                                                                                                                        |
| ttti     | int    | WSAGetServiceClassInfoA(_In_ LPGUID lpProviderId, _In_ LPGUID lpServiceClassId, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSASERVICECLASSINFO lpServiceClassInfo)                                                                                                                       |
| ttti     | int    | WSAGetServiceClassInfoW(_In_ LPGUID lpProviderId, _In_ LPGUID lpServiceClassId, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSASERVICECLASSINFO lpServiceClassInfo)                                                                                                                       |
| tsti     | int    | WSAGetServiceClassNameByClassId(_In_ LPGUID lpServiceClassId, _Out_ LPTSTR lpszServiceClassName, _Inout_ LPDWORD lpdwBufferLength)                                                                                                                                                      |
| tati     | int    | WSAGetServiceClassNameByClassIdA(_In_ LPGUID lpServiceClassId, _Out_ LPSTR lpszServiceClassName, _Inout_ LPDWORD lpdwBufferLength)                                                                                                                                                      |
| twti     | int    | WSAGetServiceClassNameByClassIdW(_In_ LPGUID lpServiceClassId, _Out_ LPWSTR lpszServiceClassName, _Inout_ LPDWORD lpdwBufferLength)                                                                                                                                                     |
| tuiti    | int    | WSAHost(_In_ SOCKET s, _In_ u_long hostlong, _Out_ u_long *lpnetlong)                                                                                                                                                                                                                   |
| tuhti    | int    | WSAHosts(_In_ SOCKET s, _In_ u_short hostshort, _Out_ u_short *lpnetshort)                                                                                                                                                                                                              |
| ti       | int    | WSAInstallServiceClass(_In_ LPWSASERVICECLASSINFO lpServiceClassInfo)                                                                                                                                                                                                                   |
| ti       | int    | WSAInstallServiceClassA(_In_ LPWSASERVICECLASSINFO lpServiceClassInfo)                                                                                                                                                                                                                  |
| ti       | int    | WSAInstallServiceClassW(_In_ LPWSASERVICECLASSINFO lpServiceClassInfo)                                                                                                                                                                                                                  |
| tuituiti | int    | WSAIoctl(_In_ SOCKET s, _In_ DWORD dwIoControlCode, _In_ LPVOID lpvInBuffer, _In_ DWORD cbInBuffer, _Out_ LPVOID lpvOutBuffer, _In_ DWORD cbOutBuffer, _Out_ LPDWORD lpcbBytesReturned, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine) |
| ttittuit | SOCKET | WSAJoinLeaf(_In_ SOCKET s, _In_ const struct sockaddr *name, _In_ int namelen, _In_ LPWSABUF lpCallerData, _Out_ LPWSABUF lpCalleeData, _In_ LPQOS lpSQOS, _In_ LPQOS lpGQOS, _In_ DWORD dwFlags)                                                                                       |

|            |     |                                                                                                                                                                                                                                                                                  |
|------------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tuiti      | int | WSALookupServiceBegin(_In_ LPWSAQUERYSET lpqsRestrictions, _In_ DWORD dwControlFlags, _Out_ LPHANDLE lphLookup)                                                                                                                                                                  |
| tuiti      | int | WSALookupServiceBeginA(_In_ LPWSAQUERYSET lpqsRestrictions, _In_ DWORD dwControlFlags, _Out_ LPHANDLE lphLookup)                                                                                                                                                                 |
| tuiti      | int | WSALookupServiceBeginW(_In_ LPWSAQUERYSET lpqsRestrictions, _In_ DWORD dwControlFlags, _Out_ LPHANDLE lphLookup)                                                                                                                                                                 |
| ti         | int | WSALookupServiceEnd(_In_ HANDLE hLookup)                                                                                                                                                                                                                                         |
| tuitti     | int | WSALookupServiceNext(_In_ HANDLE hLookup, _In_ DWORD dwControlFlags, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSAQUERYSET lpqsResults)                                                                                                                                          |
| tuitti     | int | WSALookupServiceNextA(_In_ HANDLE hLookup, _In_ DWORD dwControlFlags, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSAQUERYSET lpqsResults)                                                                                                                                         |
| tuitti     | int | WSALookupServiceNextW(_In_ HANDLE hLookup, _In_ DWORD dwControlFlags, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPWSAQUERYSET lpqsResults)                                                                                                                                         |
| tuiti      | int | WSANetbi(_In_ SOCKET s, _In_ u_long netlong, _Out_ u_long *lphostlong)                                                                                                                                                                                                           |
| tuhti      | int | WSANetbs(_In_ SOCKET s, _In_ u_short netshort, _Out_ u_short *lphostshort)                                                                                                                                                                                                       |
| tuiii      | int | WSAPoll(_Inout_ WSAPOLLFD fdarray[], _In_ ULONG nfds, _In_ INT timeout)                                                                                                                                                                                                          |
| tii        | int | WSAProviderCompleteAsyncCall(HANDLE hAsyncCall, INT iRetCode)                                                                                                                                                                                                                    |
| ttti       | int | WSAProviderConfigChange(_Inout_ LPHANDLE lpNotificationHandle, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine)                                                                                                                   |
| ttuittiti  | int | WSARecv(_In_ SOCKET s, _Inout_ LPWSABUF lpBuffers, _In_ DWORD dwBufferCount, _Out_ LPDWORD lpNumberOfBytesRecv, _Inout_ LPDWORD lpFlags, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine)                                         |
| titi       | int | WSARecvDisconnect(_In_ SOCKET s, _Out_ LPWSABUF lpInboundDisconnectData)                                                                                                                                                                                                         |
| ttuitttiti | int | WSARecvFrom(_In_ SOCKET s, _Inout_ LPWSABUF lpBuffers, _In_ DWORD dwBufferCount, _Out_ LPDWORD lpNumberOfBytesRecv, _Inout_ LPDWORD lpFlags, _Out_ struct sockaddr *lpFrom, _Inout_ LPINT lpFromlen, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE) |

|             |        |                                                                                                                                                                                                                                                                                       |
|-------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |        | lpCompletionRoutine)                                                                                                                                                                                                                                                                  |
| ti          | int    | WSARemoveServiceClass(_In_ LPGUID lpServiceClassId)                                                                                                                                                                                                                                   |
| ti          | BOOL   | WSAResetEvent(_In_ WSAEVENT hEvent)                                                                                                                                                                                                                                                   |
| ttuituiti   | int    | WSASend(_In_ SOCKET s, _In_ LPWSABUF lpBuffers, _In_ DWORD dwBufferCount, _Out_ LPDWORD lpNumberOfBytesSent, _In_ DWORD dwFlags, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine)                                                      |
| tii         | int    | WSASendDisconnect(_In_ SOCKET s, _In_ LPWSABUF lpOutboundDisconnectData)                                                                                                                                                                                                              |
| ttuittti    | int    | WSASendMsg(_In_ SOCKET s, _In_ LPWSAMSG lpMsg, _In_ DWORD dwFlags, _Out_ LPDWORD lpNumberOfBytesSent, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine)                                                                                 |
| ttuituititi | int    | WSASendTo(_In_ SOCKET s, _In_ LPWSABUF lpBuffers, _In_ DWORD dwBufferCount, _Out_ LPDWORD lpNumberOfBytesSent, _In_ DWORD dwFlags, _In_ const struct sockaddr *lpTo, _In_ int iToLen, _In_ LPWSAOVERLAPPED lpOverlapped, _In_ LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine) |
| ti          | BOOL   | WSASetEvent(_In_ WSAEVENT hEvent)                                                                                                                                                                                                                                                     |
| ii          | VOID   | WSASetLastError(_In_ int iError)                                                                                                                                                                                                                                                      |
| tuiiii      | int    | WSASetService(_In_ LPWSAQUERYSET lpqsRegInfo, _In_ WSAESETSERVICEOP essOperation, _In_ DWORD dwControlFlags)                                                                                                                                                                          |
| tuiiii      | int    | WSASetServiceA(_In_ LPWSAQUERYSET lpqsRegInfo, _In_ WSAESETSERVICEOP essOperation, _In_ DWORD dwControlFlags)                                                                                                                                                                         |
| tuiiii      | int    | WSASetServiceW(_In_ LPWSAQUERYSET lpqsRegInfo, _In_ WSAESETSERVICEOP essOperation, _In_ DWORD dwControlFlags)                                                                                                                                                                         |
| iiituiuit   | SOCKET | WSASocket(_In_ int af, _In_ int type, _In_ int protocol, _In_ LPWSAPROTOCOL_INFO lpProtocolInfo, _In_ GROUP g, _In_ DWORD dwFlags)                                                                                                                                                    |
| iiituiuit   | SOCKET | WSASocketA(_In_ int af, _In_ int type, _In_ int protocol, _In_ LPWSAPROTOCOL_INFO lpProtocolInfo, _In_ GROUP g, _In_ DWORD dwFlags)                                                                                                                                                   |
| iiituiuit   | SOCKET | WSASocketW(_In_ int af, _In_ int type, _In_ int protocol, _In_ LPWSAPROTOCOL_INFO lpProtocolInfo, _In_ GROUP g, _In_ DWORD dwFlags)                                                                                                                                                   |
| uhti        | int    | WSAStartup(_In_ WORD wVersionRequested, _Out_ LPWSADATA lpWSAData)                                                                                                                                                                                                                    |
|             |        |                                                                                                                                                                                                                                                                                       |

|             |       |                                                                                                                                                                                                                          |
|-------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sittti      | int   | WSAStringToAddress(_In_ LPTSTR AddressString, _In_ INT AddressFamily, _In_opt_ LPWSAPROTOCOL_INFO lpProtocolInfo, _Out_ LPSOCKADDR lpAddress, _Inout_ LPINT lpAddressLength)                                             |
| aittti      | int   | WSAStringToAddressA(_In_ LPSTR AddressString, _In_ INT AddressFamily, _In_opt_ LPWSAPROTOCOL_INFO lpProtocolInfo, _Out_ LPSOCKADDR lpAddress, _Inout_ LPINT lpAddressLength)                                             |
| wittti      | int   | WSAStringToAddressW(_In_ LPWSTR AddressString, _In_ INT AddressFamily, _In_opt_ LPWSAPROTOCOL_INFO lpProtocolInfo, _Out_ LPSOCKADDR lpAddress, _Inout_ LPINT lpAddressLength)                                            |
| ti          | int   | WSAUnadvertiseProvider(_In_ const GUID *puuidProviderId)                                                                                                                                                                 |
| uitiuui     | DWORD | WSAWaitForMultipleEvents(_In_ DWORD cEvents, _In_ const WSAEVENT *lphEvents, _In_ BOOL fWaitAll, _In_ DWORD dwTimeout, _In_ BOOL fAlertable)                                                                             |
| tii         | int   | WSCDeinstallProvider(_In_ LPGUID lpProviderId, _Out_ LPINT lpErrno)                                                                                                                                                      |
| tiii        | int   | WSCEnableNSProvider(_In_ LPGUID lpProviderId, _In_ BOOL fEnable)                                                                                                                                                         |
| tttti       | int   | WSCEnumProtocols(_In_ LPINT lpiProtocols, _Out_ LPWSAPROTOCOL_INFOW lpProtocolBuffer, _Inout_ LPDWORD lpdwBufferLength, _Out_ LPINT lpErrno)                                                                             |
| wuiwuitti   | int   | WSCGetApplicationCategories(_In_ LPCWSTR Path, _In_ DWORD PathLength, _In_ LPCWSTR Extra, _In_ DWORD ExtraLength, _Out_ DWORD *pPermittedLspCategories, _Out_ LPINT lpErrno)                                             |
| tuittuiti   | int   | WSCGetProviderInfo(_In_ LPGUID lpProviderId, _In_ WSC_PROVIDER_INFO_TYPE InfoType, _Out_ PBYTE Info, _Inout_size_t *InfoSize, _In_ DWORD Flags, _Out_ LPINT lpErrno)                                                     |
| twtti       | int   | WSCGetProviderPath(_In_ LPGUID lpProviderId, _Out_ LPWSTR lpszProviderDllPath, _Inout_ LPINT lpProviderDllPathLen, _Out_ LPINT lpErrno)                                                                                  |
| wwuiiuiti   | int   | WSCInstallNameSpace(_In_ LPWSTR lpszIdentifier, _In_ LPWSTR lpszPathName, _In_ DWORD dwNameSpace, _In_ DWORD dwVersion, _In_ LPGUID lpProviderId)                                                                        |
| wwuiiuitti  | int   | WSCInstallNameSpaceEx(_In_ LPWSTR lpszIdentifier, _In_ LPWSTR lpszPathName, _In_ DWORD dwNameSpace, _In_ DWORD dwVersion, _In_ LPGUID lpProviderId, _In_ LPBLOB lpProviderInfo)                                          |
| twtuiti     | int   | WSCInstallProvider(_In_ const LPGUID lpProviderId, _In_ const LPWSTR lpszProviderDllPath, _In_ const LPWSAPROTOCOL_INFO lpProtocolInfoList, _In_ DWORD dwNumberOfEntries, _Out_ LPINT lpErrno)                           |
| twwuituitti | int   | WSCInstallProviderAndChain(_In_ const LPGUID lpProviderId, _In_ const LPWSTR lpszProviderDllPath, _In_ const LPWSTR lpszLspName, _In_ DWORD dwServiceFlags, _In_ const LPWSAPROTOCOL_INFO lpProtocolInfoList, _In_ DWORD |

|            |     |                                                                                                                                                                                                       |
|------------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |     | dwNumberOfEntries, _Out_opt_ LPDWORD lpdwCatalogEntryId, _Out_ LPINT lpErrno)                                                                                                                         |
| wuiwuiuiti | int | WSCSetApplicationCategory(_In_ LPCWSTR Path, _In_ DWORD PathLength, _In_ LPCWSTR Extra, _In_ DWORD ExtraLength, _In_ DWORD PermittedLspCategories, _Out_ DWORD *pPrevPermLspCat, _Out_ LPINT lpErrno) |
| tuittuiti  | int | WSCSetProviderInfo(_In_ LPGUID lpProviderId, _In_ WSC_PROVIDER_INFO_TYPE InfoType, _In_ PBYTE Info, _In_ size_t InfoSize, _In_ DWORD Flags, _Out_ LPINT lpErrno)                                      |
| ti         | int | WSCUninstallNameSpace(_In_ LPGUID lpProviderId)                                                                                                                                                       |
| twtuiti    | int | WSCUpdateProvider(_In_ LPGUID lpProviderId, _In_ const WCHAR *lpszProviderDllPath, _In_ const LPWSAPROTOCOL_INFO lpProtocolInfoList, _In_ DWORD dwNumberOfEntries, _Out_ LPINT lpErrno)               |

# ComObjActive()

Retrieves a running object that has been registered with OLE.

```
OutputVar := ComObjActive(CLSID)
```

```
Function Example: aOut :=
ComObjActive("Outlook.Application")
```

## Parameters

### OutputVar

The name of the variable in which to store a COM wrapper object which can be used with [object syntax](#).

Default behaviour. [AddRef](#) is called automatically for IUnknown and IDispatch pointers, so the caller should use [ObjRelease](#) to release their copy of the pointer if appropriate.

As the default behaviour may be changing in a future release, it is recommended to always set *Flags* to [1](#) when wrapping an interface pointer, and call [ObjAddRef](#) if needed.

### CLSID

CLSID or human-readable Prog ID of the COM object to retrieve.

## Related

[ComObject](#), [ComObjCreate](#), [ComObjGet](#), [ComObjConnect](#), [ComObjError](#), [ComObjFlags](#), [ObjAddRef/ObjRelease](#), [ComObjQuery](#), [GetActiveObject](#) (MSDN)

## Example

```
; Displays the active document in Microsoft Word,
if it is running.
```

```
word := ComObjActive("Word.Application")
if !word
 MsgBox Word isn't open.
else
 MsgBox % word.ActiveDocument.FullName
```

# ComObjArray

Creates a SafeArray for use with COM.

```
OutputVar := ComObjArray(VarType, Count1 [, Count2,
... Count8])
```

```
Function Example: arr := ComObjArray(VT_BSTR :=
8, 10, 10, 10)
```

## Parameters

### OutputVar

The name of the variable in which to store the wrapper object containing the SafeArray.

### VarType

The base type of the array (the VARTYPE of each element of the array). The VARTYPE is restricted to a subset of the variant types. Neither the VT\_ARRAY nor the VT\_BYREF flag can be set. VT\_EMPTY and VT\_NULL are not valid base types for the array. All other types are legal. See [ComObjType](#) for a list of possible values.

### CountN

The size of each dimension. Arrays containing up to 8 dimensions are supported.

## ArrayObj

A wrapper object containing the SafeArray.

## Methods

Array wrapper objects support the following methods:

`Array.MaxIndex(n)`: Returns the upper bound of the *n*th dimension. If *n* is omitted, it defaults to 1.

`Array.MinIndex(n)`: Returns the lower bound of the *n*th dimension. If *n* is omitted, it defaults to 1.

`Array.Clone()`: Returns a copy of the array.

`Array._NewEnum()`: Not typically called by script; allows [for-loops](#) to be used with SafeArrays.

## General Remarks

Array wrapper objects may also be returned by COM methods and [ComObject](#). Scripts may determine if a value is an array as follows:

```
if ComObjType(obj) & 0x2000
 MsgBox % "Array subtype: " .
ComObjType(obj) & 0xfff
else
 MsgBox Not an array.
```

Arrays with up to 8 dimensions are supported.

Since SafeArrays are not designed to support multiple references, when one SafeArray is assigned to an element of another SafeArray, a separate copy is created. However, this only occurs if the wrapper object has the `F_OWNVALUE` flag, which indicates it is responsible for destroying the array. This flag can be removed by using [ComObjFlags](#).

When a function or method called by a COM client returns a SafeArray with the `F_OWNVALUE` flag, a copy is created and returned instead, as the original SafeArray is automatically destroyed.

## Related

[ComObject](#), [ComObjType](#), [ComObjValue](#), [ComObjActive](#), [ComObjFlags](#), [Array Manipulation Functions \(MSDN\)](#)

## Examples

**; Example #1: Simple usage.**

```
arr := ComObjArray(VT_VARIANT:=12, 3)
arr[0] := "Auto"
arr[1] := "Hot"
arr[2] := "key"
Loop % arr.MaxIndex() + 1
 t := arr[A_Index-1]
MsgBox % t
```

**; Example #2: Multiple dimensions.**

```
arr := ComObjArray(VT_VARIANT:=12, 3, 4)
```

```
; Get the number of dimensions:
```

```
dim := DllCall("oleaut32\SafeArrayGetDim", "ptr",
ComObjValue(arr))
```

```
; Get the bounds of each dimension:
```

```
Loop dim
 dims .= arr.MinIndex(A_Index) " .. "
arr.MaxIndex(A_Index) "`n"
MsgBox %dims%
```

```
; Simple usage:
```

```
Loop 3 {
 x := A_Index-1
 Loop 4 {
 y := A_Index-1
 arr[x, y] := x * y
 }
}
MsgBox % arr[2, 3]
```

# ComObjConnect

Connects the object's event sources to functions with a given prefix.

```
ComObjConnect ComObject [, Prefix]
```

```
Command Example: ComObjConnect ie, "IE_"
Function Example: ComObjConnect(ie, "IE_")
```

## Parameters

### ComObject

An object which raises events.

If the object does not support the `IConnectionPointContainer` interface or type information about the object's class cannot be retrieved, an error message is shown. This can be suppressed or handled with [ComObjError](#) or `try/catch`.

The `IProvideClassInfo` interface is used to retrieve type information about the object's class if the object supports it. Otherwise, `ComObjConnect` attempts to retrieve type information via the object's `IDispatch` interface, which may be unreliable.

### Prefix

A string to prefix to the event name to determine which function to call when an event occurs.

If omitted, the object is "disconnected"; that is, the script will no longer receive notification of its events.

This parameter can be an object defined by the script. When an event is raised, the corresponding method is called. The first parameter, which is usually the hidden `this` parameter, refers to the script-defined object, not the COM object. To catch all events without defining a method for each one, define a `__Call` meta-function.

## Usage

To make effective use of `ComObjConnect`, you must first write functions in the script to handle any events of interest. Such functions, or "event-handlers," have the following structure:

```
PrefixEventName([Params..., ComObject])
{
 .. event-handling code ..
 return returnValue
}
```

*Prefix* is a prefix of your choosing, while **EventName** is the name of whatever event the function should handle.

*Params* corresponds to whatever parameters the event has. If the event has no parameters, *Params* should be omitted entirely. *ComObject* is optional, and can only be used if the correct number of *Params* are defined; it contains a reference to the original wrapper object which was passed to `ComObjConnect`.

"ComObject" should be replaced with a name more meaningful in the context of your script.

Note that event handlers may have return values. To return a COM-specific type of value, use `ComObject(type, value)`. For example, `return ComObject(0, 0)` returns a variant of type VT\_EMPTY, which is equivalent to returning `undefined` (or not returning) from a JavaScript function.

Call `ComObjConnect(yourObject, "Prefix")` to enable event-handling.

Call `ComObjConnect(yourObject)` to disconnect the object (stop handling events).

If the number of parameters is not known, a [variadic function](#) can be used.

## Related

[ComObjCreate](#), [ComObjGet](#), [ComObjActive](#), [ComObjError](#),  
[WScript.ConnectObject](#) (MSDN)

## Examples

```
ie := ComObjCreate("InternetExplorer.Application")
; Connects events to corresponding script
functions with the prefix "IE_".
ComObjConnect(ie, "IE_")

ie.Visible := true ; This is known to work
```

**incorrectly on IE7.**

```
ie.Navigate("https://autohotkey.com/")
#Persistent
```

```
IE_DocumentComplete(ieEventParam, url,
ieFinalParam) {
 global ie
 if (ie != ieEventParam)
 s .= "First parameter is a new wrapper
object.\`n"
 if (ie == ieFinalParam)
 s .= "Final parameter is the original
wrapper object.\`n"
 if (ComObjValue(ieEventParam) ==
ComObjValue(ieFinalParam))
 s .= "Both wrapper objects refer to the
same IDispatch instance.\`n"
 MsgBox % s . "Finished loading "
ie.Document.title " @ " url
 ie.Quit()
 ExitApp
}
```

# ComObjCreate

Creates a COM object.

```
OutputVar := ComObjCreate(CLSID [, IID])
```

```
Function Example: ie :=
ComObjCreate("InternetExplorer.Application")
```

## Parameters

### OutputVar

The name of the variable in which to store the result. The result is different when IID parameter is used:

- If **IID** is not specified a wrapper object supporting [object syntax](#) is returned.
- When **IID** is specified, interface pointer is returned. The script must typically call [ObjRelease](#) when it is finished with the pointer.

If an error occurs, an empty string is returned.

### CLSID

CLSID or human-readable Prog ID of the COM object to create.

### IID

The identifier of an interface the object supports.

## Related

[ComObject](#), [ComObjGet](#), [ComObjActive](#), [ComObjConnect](#), [ComObjArray](#), [ComObjError](#), [ComObjQuery](#), [CreateObject \(MSDN\)](#)

## Examples

For a constantly growing list of examples, see the following forum topic:

<http://www.autohotkey.com/forum/topic61509.html>.

```
ie := ComObjCreate("InternetExplorer.Application")
ie.Visible := true ; This is known to work
incorrectly on IE7.
ie.Navigate("https://autohotkey.com/")
```

# ComObjDll

Creates a COM object from a dll.

```
OutputVar := ComObjDll(hModule, CLSID [, IID])
```

```
Function Example: ahk := ComObjDll(hModule, "{C00BCC8C-5A04-4392-870F-20AAE1B926B2}")
```

## Parameters

### OutputVar

The name of the variable in which to store the result. The result is different when IID parameter is used:

- If **IID** is not specified a wrapper object supporting [object syntax](#) is returned.
- When **IID** is specified, interface pointer is returned. The script must typically call [ObjRelease](#) when it is finished with the pointer.

If an error occurs, an empty string is returned.

### hModule

Dll Module handle, loaded via [LoadLibrary](#) or [MemoryLoadLibrary](#).

### CLSID

CLSID or human-readable Prog ID of the COM object to create.

## **IID**

The identifier of an interface the object supports.

## **Related**

[ComObjCreate](#) [ComObjGet](#) [ComObjActive](#) [ComObjConnect](#) [ComObjArray](#)  
[ComObjError](#) [ComObjQuery](#) [CreateObject \(MSDN\)](#)

## **Examples**

For a constantly growing list of examples, see the following forum topic:  
<http://www.autohotkey.com/forum/topic61509.html>.

```
ie := ComObjCreate("InternetExplorer.Application")
ie.Visible := true ; This is known to work
incorrectly on IE7.
ie.Navigate("http://1.autohotkey.net/")
```

# ComObjGet

Returns a reference to an object provided by a COM component.

```
OutputVar := ComObjGet(Name)
```

```
Function Example: wmi := ComObjGet("winmgmts:")
```

## Parameters

### OutputVar

The name of the variable in which to store the Com object.

### Name

The display name of the object to be retrieved. See [MkParseDisplayName \(MSDN\)](#) for more information.

## Related

[ComObjCreate](#) [ComObjActive](#) [ComObjConnect](#) [ComObjError](#)

[ComObjQuery](#) [CoGetObject \(MSDN\)](#)

## Examples

```
; Example: Press Shift+Escape to show the command
line which was used
; to launch the active window's process.
```

Requires XP or later.

+Esc: :

```
pid := WinGetPID("A")
; Get WMI service object.
wmi := ComObjGet("winmgmts:")
; Run query to retrieve matching process(es).
queryEnum := wmi.ExecQuery("
 . "Select * from Win32_Process where
ProcessId=" . pid)
 ._NewEnum()
; Get first matching process.
if queryEnum[process]
 MsgBox(process.CommandLine, "Command
line", 0)
else
 MsgBox Process not found!
; Free all global objects (not necessary when
using local vars).
wmi := queryEnum := process := ""
return
; Win32_Process: http://msdn.microsoft.com/en-
us/library/aa394372.aspx
```

# ComObject

Wraps a value, SafeArray or COM object for use by the script or for passing to a COM method.

```
ComObject := ComObject(VarType, Value [, Flags])
```

**Advanced:** Wraps or unwraps a raw [IDispatch](#) pointer for use by the script.

```
ComObject := ComObject(DispPtr)
```

## Parameters

### VarType

An integer indicating the type of value. See [ComObjType](#) for a list of types.

### Value

The value to wrap. Currently only integer and pointer values are supported.

### Flags

Flags affecting the behaviour of the wrapper object; see [ComObjFlags](#) for details.

### DispPtr

Raw IDispatch pointer.

## Return Value

Returns a wrapper object containing a [variant type](#) and value or pointer.

This object has two uses:

1. Some COM methods may require specific types of values which have no direct equivalent within AutoHotkey. This function allows the type of a value to be specified when passing it to a COM method. For example, `ComObject(0xB, -1)` creates an object which represents the COM boolean value *true*.
2. Wrapping a COM object or SafeArray enables the script to interact with it more naturally, using [object syntax](#). However, the majority of scripts do not need to do this manually since a wrapper object is created automatically by [ComObjCreate](#), [ComObjArray](#) and any COM methods which returns an object.

## ByRef

If a wrapper object's *VarType* includes the VT\_BYREF (0x4000) flag, empty brackets `[]` can be used to read or write the referenced value.

When creating a reference, *Value* must be the memory address of a variable or buffer with sufficient capacity to store a value of the given type. For example, the following can be used to create a variable which a VBScript function can write into:

```
VarSetCapacity(var, 24, 0)
vref := ComObject(0x400C, &var) ; 0x400C is a
combination of VT_BYREF and VT_VARIANT.

vref[] := "in value"
sc.Run("Example", vref) ; sc should be
initialized as in the example below.
MsgBox % vref[]
```

## General Remarks

When this function is used to wrap an [IDispatch](#) or [IUnknown](#) interface pointer, the wrapper object assumes responsibility for automatically releasing the pointer when appropriate. If *VarType* was omitted, the object is [queried](#) for its [IDispatch](#) interface; if one is returned, *DispPtr* is immediately released. Therefore, if the script intends to use the pointer after calling this function, it must call [ObjAddRef](#)([DispPtr](#)) first.

The *VarType* of a wrapper object can be retrieved using [ComObjType](#).

The *Value* of a wrapper object can be retrieved using [ComObjValue](#).

**Known limitation:** Each time a COM object is wrapped, a new wrapper object is created. Comparisons and assignments such as `obj1 == obj2` and `array[obj1] := value` treat the two wrapper objects as unique, even though they contain the same COM object.

## Related

ComObjCreate, ComObjGet, ComObjConnect, ComObjError, ComObjFlags, ObjAddRef/ObjRelease, ComObjQuery, GetActiveObject (MSDN)

## Examples

```
; Preamble - ScriptControl requires a 32-bit version of AutoHotkey.
```

```
code := "(
Sub Example(Var)
 MsgBox Var
 Var = "out value!"
End Sub
)"
sc := ComObjCreate("ScriptControl"), sc.Language := "VBScript", sc.AddCode(code)
```

```
; Example: Pass a VARIANT ByRef to a COM function.
```

```
var := ComVar()
var[] := "in value"
sc.Run("Example", var.ref)
MsgBox % var[]
```

```
; ComVar: Creates an object which can be used to pass a value ByRef.
```

```
; ComVar[] retrieves the value.
```

```
; ComVar[] := Val sets the value.
```

```
; ComVar.ref retrieves a ByRef object for passing to a COM function.
```

```
ComVar(Type:=0xC)
{
 static base := { __Get: "ComVarGet", __Set: "ComVarSet", __Delete: "ComVarDel" }
```

```
; Create an array of 1 VARIANT. This method
```

```

allows built-in code to take
; care of all conversions between VARIANT and
AutoHotkey internal types.
arr := ComObjArray(Type, 1)
; Lock the array and retrieve a pointer to the
VARIANT.
DllCall("oleaut32\SafeArrayAccessData", "ptr",
ComObjValue(arr), "ptr*", arr_data)
; Store the array and an object which can be
used to pass the VARIANT ByRef.
return { ref: ComObject(0x4000|Type,
arr_data), _: arr, base: base }
}
ComVarGet(cv, p*) { ; Called when script accesses
an unknown field.
if !p.Length() ; No name/parameters, i.e.
cv[]
return cv._[0]
}
ComVarSet(cv, v, p*) { ; Called when script sets
an unknown field.
if !p.Length() ; No name/parameters, i.e.
cv[]:=v
return cv._[0] := v
}
ComVarDel(cv) { ; Called when the object is being
freed.
; This must be done to allow the internal
array to be freed.
DllCall("oleaut32\SafeArrayUnaccessData",
"ptr", ComObjValue(cv._))
}

```

# ComObjError()

Enables or disables notification of COM errors.

```
Enabled := ComObjError([Enable])
```

## Parameters

### Enable

A boolean value (true or false). Optional.

### Enabled

Receives the setting which was in effect before the function was called.

## General Remarks

After accessing a COM object, `A_LastError` contains the HRESULT code returned by the COM object's `IDispatch.Invoke` function. Scripts may implement their own error-handling by calling `ComObjError(false)` to disable error notifications and consulting the value of `A_LastError`.

## Related

[ComObjCreate](#), [ComObjGet](#), [ComObjActive](#), [ComObjConnect](#)

# ComObjFlags

Retrieves or changes flags which control a COM wrapper object's behaviour.

```
OutputVar := ComObjFlags([NewFlags, Mask])
```

```
Command Example: ComObjFlags ComObject, -1
Function Example: flags :=
ComObjFlags(ComObject, -1)
```

## Parameters

### OutputVar

The name of the variable in which to store all *ComObject*'s flags (after applying *NewFlags*, if specified).

### ComObject

A COM wrapper object.

### NewFlags (optional)

New values for the flags identified by *Mask*, or flags to add or remove.

### Mask (optional)

A bitmask of flags to change.

### Flags

All of *ComObject*'s flags (after applying *NewFlags*, if specified).

## Flags

F\_OWNVALUE  
1

Currently only affects SafeArrays. If this flag is set, the SafeArray is destroyed when the wrapper object is freed. Since SafeArrays have no reference counting mechanism, if a SafeArray with this flag is assigned to an element of another SafeArray, a separate copy is created.

## General Remarks

If *Mask* is omitted, *NewFlags* specifies the flags to add (if positive) or remove (if negative). For example, `ComObjFlags(obj, -1)` removes the F\_OWNVALUE flag. Do not specify any value for *Mask* other than 0 or 1; all other bits are reserved for future use.

## Related

[ComObject](#), [ComObjActive](#), [ComObjArray](#)

## Examples

```
; Example: Check for the presence of the
F_OWNVALUE flag.
```

```
arr := ComObjArray(0xC, 1)
if ComObjFlags(arr) & 1
 MsgBox arr will be automatically destroyed.
else
 MsgBox arr will not be automatically
 destroyed.
```

```
; Example: Change array-in-array behaviour.
```

```
arr1 := ComObjArray(0xC, 3)
arr2 := ComObjArray(0xC, 1)
arr2[0] := "original value"
arr1[0] := arr2 ; Assign implicit copy.
ComObjFlags(arr2, -1) ; Remove F_OWNVALUE.
arr1[1] := arr2 ; Assign original array.
arr1[2] := arr2.Clone() ; Assign explicit copy.
arr2[0] := "new value"
for arr in arr1
 MsgBox % arr[0]
```

```
arr1 := ""
```

```
; Not valid since arr2 == arr1[1], which has been
destroyed:
```

```
; arr2[0] := "foo"
```

# ComObjQuery

Queries a COM object for an interface or service.

```
OutputVar := ComObjQuery([SID], IID)
```

```
Function Example: InterfacePointer :=
ComObjQuery(ComObject, "{332C4427-26CB-11D0-
B483-00C04FD90119}", "{332C4427-26CB-11D0-B483-
00C04FD90119}")
```

## Parameters

### OutputVar

The name of the variable in which to store the interface pointer.

### ComObject

A COM wrapper object or raw interface pointer.

### IID

An interface identifier (GUID) in the form "{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}".

### SID

A service identifier in the same form as IID. When omitting this parameter, also omit the comma.

## General Remarks

In its two-parameter mode, this function is equivalent to [IUnknown::QueryInterface](#). When SID and IID are both specified, it internally queries for the [IServiceProvider](#) interface, then calls [IServiceProvider::QueryService](#). In either case, the return value is either zero or a pointer to the requested interface. Generally this pointer must be [released](#) when the script is finished with it.

## Related

[ObjRelease](#) [ComObjCreate](#) [ComObjGet](#) [ComObjActive](#) [ComObjError](#)

## Examples

```
; Example: Determine the class name of an object.
```

```
obj := ComObjCreate("Scripting.Dictionary")

MsgBox % "Interface name: " ComObjType(obj,
"name")

IID_IProvideClassInfo := "{B196B283-BAB4-101A-
B69C-00AA00341D07}"
```

```
; Request a pointer to the object's
IProvideClassInfo interface.
```

```
if !(pci := ComObjQuery(obj,
IID_IProvideClassInfo))
{
 MsgBox IProvideClassInfo interface not
```

```
supported.
 return
}
```

```
; Call GetClassInfo to retrieve a pointer to the
ITypeInfo interface.
```

```
DllCall(vtable(pci, 3), "ptr", pci, "ptr*", ti)
```

```
; Call GetDocumentation to get the object's full
type name.
```

```
DllCall(vtable(ti, 12), "ptr", ti, "int", -1,
"ptr*", name, "ptr", 0, "ptr", 0, "ptr", 0)
```

```
; Convert the BSTR pointer to a usable string.
```

```
name := StrGet(name, "UTF-16")
```

```
; Release raw interface pointers.
```

```
ObjRelease(ti)
ObjRelease(pci)
```

```
; Display the type name!
```

```
MsgBox % "Class name: " name
```

```
vtable(ptr, n) {
```

```
 ; NumGet(ptr+0) returns the address of the
 object's virtual function
```

```
 ; table (vtable for short). The remainder of
 the expression retrieves
```

```
 ; the address of the nth function's address
 from the vtable.
```

```
 return NumGet(NumGet(ptr+0), n*A_PtrSize)
}
```

```
; Example: Automate an existing Internet Explorer
window.
```

```

sURL := "https://autohotkey.com/boards/"
if webBrowser := GetWebBrowser()
 webBrowser.Navigate(sURL)
return

GetWebBrowser()
{
 ; Get a raw pointer to the document object of
 the top-most IE window.
 static msg := DllCall("RegisterWindowMessage",
 "str", "WM_HTML_GETOBJECT")
 SendMessage, %msg%, 0, 0, Internet
 Explorer_Server1, ahk_class IEFrame
 if ErrorLevel = "ERROR"
 return ; IE not found.
 lResult := ErrorLevel
 DllCall("oleacc\ObjectFromLresult", "ptr",
 lResult
 , "ptr", GUID(IID_IHTMLDocument2,
 {332C4425-26CB-11D0-B483-00C04FD90119})
 , "ptr", 0, "ptr*", pdoc)

 ; Query for the WebBrowserApp service. In this
 particular case,
 ; the SID and IID are the same, but it isn't
 always this way.
 static IID_IWebBrowserApp := "{0002DF05-0000-
 0000-C000-000000000046}"
 static SID_SWebBrowserApp :=
 IID_IWebBrowserApp
 pweb := ComObjQuery(pdoc, SID_SWebBrowserApp,
 IID_IWebBrowserApp)

 ; Release the document object pointer.
 ObjRelease(pdoc)

 ; Return the WebBrowser object, wrapped for

```

### usability:

```
static VT_DISPATCH := 9, F_OWNVALUE := 1
return ComObject(VT_DISPATCH, pweb,
F_OWNVALUE)
}
```

GUID(ByRef GUID, sGUID) ; Converts a string to a binary GUID and returns its address.

```
{
 VarSetCapacity(GUID, 16, 0)
 return DllCall("ole32\CLSIDFromString",
"wstr", sGUID, "ptr", &GUID) >= 0 ? &GUID : ""
}
```

# ComObjType

Retrieves type information from a COM object.

```
OutputVar := ComObjType(ComObject [, Value])
```

```
Function Example: IID :=
ComObjType(ComObject, "Name")
```

## Parameters

### OutputVar

The name of the variable in which to store the result depending on Parameter, see **Value**.

### ComObject

A wrapper object containing a COM object or typed value.

### Value (optional)

The result will differ depending on this parameter:

- **VarType** - Omit the **Value** parameter to retrieve the type of value.
- **Name** - Pass "Name" in **Value** parameter to retrieve the interface type name.
- **IID** - Pass "IID" in **Value** parameter to retrieve a globally unique identifier (GUID) representing the interface type.

## Variant Type Constants

```
VT_EMPTY := 0 ; No value
VT_NULL := 1 ; SQL-style Null
VT_I2 := 2 ; 16-bit signed int
VT_I4 := 3 ; 32-bit signed int
VT_R4 := 4 ; 32-bit floating-point
number
VT_R8 := 5 ; 64-bit floating-point
number
VT_CY := 6 ; Currency
VT_DATE := 7 ; Date
VT_BSTR := 8 ; COM string (Unicode
string with length prefix)
VT_DISPATCH := 9 ; COM object
VT_ERROR := 0xA ; Error code (32-bit
integer)
VT_BOOL := 0xB ; Boolean True (-1) or
False (0)
VT_VARIANT := 0xC ; VARIANT (must be
combined with VT_ARRAY or VT_BYREF)
VT_UNKNOWN := 0xD ; IUnknown interface
pointer
VT_DECIMAL := 0xE ; (not supported)
VT_I1 := 0x10 ; 8-bit signed int
VT_UI1 := 0x11 ; 8-bit unsigned int
VT_UI2 := 0x12 ; 16-bit unsigned int
VT_UI4 := 0x13 ; 32-bit unsigned int
VT_I8 := 0x14 ; 64-bit signed int
VT_UI8 := 0x15 ; 64-bit unsigned int
VT_INT := 0x16 ; Signed machine int
VT_UINT := 0x17 ; Unsigned machine int
VT_RECORD := 0x24 ; User-defined type --
```

## NOT SUPPORTED

```
VT_ARRAY := 0x2000 ; SAFEARRAY
VT_BYREF := 0x4000 ; Pointer to another
type of value
/*
VT_ARRAY and VT_BYREF are combined with
another value (using bitwise OR)
to specify the exact type. For instance,
0x2003 identifies a SAFEARRAY
of 32-bit signed integers and 0x400C
identifies a pointer to a VARIANT.
*/
```

## General Remarks

In most common cases, return values from methods or properties of COM objects are converted to an appropriate data type supported by AutoHotkey. Types which aren't specifically handled are coerced to strings via `VariantChangeType`; if this fails or if the variant type contains the `VT_ARRAY` or `VT_BYREF` flag, an object containing both the value and its type is returned instead.

For any variable `x`, if `ComObjType(x)` returns an integer, `x` contains a COM object wrapper.

## Related

[ComObjValue](#), [ComObject](#), [ComObjCreate](#), [ComObjGet](#), [ComObjActive](#)

## Examples

---

```
d := ComObjCreate("Scripting.Dictionary")
VarType := ComObjType(d) ; Always 9 for
script-callable objects.
Name := ComObjType(d, "Name") ; Only valid for
script-callable objects.
IID := ComObjType(d, "IID") ; As above.
MsgBox Variant type:`t%VarType%`nType
name:`t%Name%`nInterface ID:`t%IID%
```

# ComObjValue

Retrieves the value or pointer stored in a COM wrapper object.

```
OutputVar := ComObjValue(ComObject)
```

```
Function Example: Value :=
ComObjValue(ComObject)
```

## Parameters

### OutputVar

The name of the variable in which to store the 64-bit signed integer.

### ComObject

A wrapper object containing a COM object or typed value.

## General Remarks

This function is not intended for general use.

Calling ComObjValue is equivalent to `variant.L1Val`, where *ComObject* is treated as a VARIANT structure. Any script which uses this function must be aware what type of value the wrapper object contains and how it should be treated. For instance, if an interface pointer is returned, `Release` should not be called, but `AddRef` may be required depending on what the script does with the pointer.

## Related

[ComObjType](#), [ComObjCreate](#), [ComObjGet](#), [ComObjActive](#)

# ObjAddRef() / ObjRelease()

Increments or decrements an object's [reference count](#).

```
OutputVar := ObjAddRef(Ptr)
OutputVar := ObjRelease(Ptr)
```

```
Function Example: RefCount := ObjAddRef(&obj;)
```

## Parameters

### OutputVar

The name of the variable in which to store the new reference count.

### Ptr

An unmanaged object pointer or COM interface pointer.

## General Remarks

Reference count should be used **only** for debugging purposes.

## Related

[Reference Counting](#)

Although the following articles discuss reference counting as it applies to COM, they cover some important concepts and rules which generally also apply to

AutoHotkey objects: [IUnknown::AddRef](#), [IUnknown::Release](#), [Reference Counting Rules](#).

## Examples

See [ComObjConnect](#).

```
obj := Object()
```

```
; The following two lines are equivalent:
```

```
ptr1 := Object(obj)
ptr2 := ObjectToPointer(obj)
```

```
ObjectToPointer(obj) {
 if !IsObject(obj)
 return ""
 ptr := &obj
 ObjAddRef(ptr)
 return ptr
}
```

```
; Each pointer retrieved via Object() or
ObjectToPointer() must be manually released
; to allow the object to be eventually freed and
any memory used by it reclaimed.
```

```
ObjRelease(ptr2)
ObjRelease(ptr1)
```

# Drive

Functions to eject/retract or lock/unlock the tray in a CD or DVD drive, or set a drive's volume label.

**Drive** Sub-command `[, Drive , Value]`

**Command Example:** Drive "Eject", "D:"  
**Function Example:** Drive("Eject", "D:")

## DriveSetLabel

Changes *Drive's* volume label to be *NewLabel*.

**DriveSetLabel** Drive `[, NewLabel]`

### Drive

The drive letter followed by a colon and an optional backslash (might also work on UNC paths and mapped drives).

### NewLabel

The new label to set. If omitted, the drive will have no label.

For example: `DriveSetLabel("C:", "Seagate200")`.

To retrieve the current label, follow this example: `Label := DriveGetLabel("C:").`

## DriveLock

Prevents a drive's eject feature from working.

**DriveLock** Drive

For example: `DriveLock("D:").`

Most drives cannot be "locked open". However, locking the drive while it is open will probably result in it becoming locked the moment it is closed.

This function has no effect on drives that do not support locking (such as most read-only drives).

To unlock a drive, call [DriveUnlock](#). If a drive is locked by a script and that script exits, the drive will stay locked until another script or program unlocks it, or the system is restarted.

If the specified drive does not exist or does not support the locking feature, [ErrorLevel](#) is set to 1. Otherwise, it is set to 0.

## DriveUnlock

Reverses DriveLock.

**DriveUnlock** Drive

For example: `DriveUnlock("D:").`

This function needs to be called multiple times if the drive was locked multiple times (at least for some drives). For example, if `DriveLock("D:")` was called three times, `DriveUnlock("D:")` might need to be called three times to unlock it. Because of this and the fact that there is no way to determine whether a drive is currently locked, it is often useful to keep track of its lock-state in a [variable](#).

## DriveEject

Ejects or retracts the tray of a CD or DVD drive.

```
DriveEject [Drive, Retract := false]
```

### Drive

The drive letter followed by a colon and an optional backslash. If omitted, the default CD/DVD drive will be used.

### Retract

Specify *true* to retract/close the tray. Specify *false* or omit this parameter to eject the tray.

`DriveEject` waits for the ejection or retraction to complete before allowing the script to continue. If the tray is already in the correct state (open or closed), [ErrorLevel](#) is set to 0 (i.e. "no error").

`DriveEject` will probably not work on a network drive or non-CD/DVD drive. If it fails in such cases or for any other reason, [ErrorLevel](#) is set to 1. To eject other

types of media or devices, see the [DllCall](#) example at the bottom of this page.

It may be possible to detect the previous tray state by measuring the time the function takes to complete. For example, the following hotkey toggles the tray to the opposite state (open or closed):

```
#c::
DriveEject()
; If the function completed quickly, the tray
was probably already ejected.
; In that case, retract it:
if A_TimeSinceThisHotkey < 1000 ; Adjust this
time if needed.
 DriveEject(, true)
return
```

To determine the media status of a CD or DVD drive (playing, stopped, open, etc.), see [DriveGetStatusCD](#).

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

These functions return 1 on success and 0 on failure.

## Remarks

The following is an alternate ejection method that also works on types of media/devices other than CD/DVD:

```
; Update the first line below to match the
desired drive letter (you can ignore all the
other lines below).
```

```
Driveletter := "I:" ; Set this to the drive
letter you wish to eject.
```

```
hVolume := DllCall("CreateFile"
 , Str, "\\.\" . Driveletter
 , UInt, 0x80000000 | 0x40000000 ;
 GENERIC_READ | GENERIC_WRITE
 , UInt, 0x1 | 0x2 ; FILE_SHARE_READ |
 FILE_SHARE_WRITE
 , UInt, 0
 , UInt, 0x3 ; OPEN_EXISTING
 , UInt, 0, UInt, 0)
if hVolume <> -1
{
 DllCall("DeviceIoControl"
 , UInt, hVolume
 , UInt, 0x2D4808 ;
 IOCTL_STORAGE_EJECT_MEDIA
 , UInt, 0, UInt, 0, UInt, 0, UInt, 0
 , UIntP, dwBytesReturned ; Unused.
 , UInt, 0)
 DllCall("CloseHandle", UInt, hVolume)
}
```

## Related

[DriveGet](#)

# DriveGet

Functions for retrieving various types of information about the computer's drive(s).

```
OutputVar := DriveGet(Cmd [, Value])
```

```
Function Example: Drives :=
DriveGet("List", "Fixed")
```

Returns a string of letters, one character for each drive letter in the system. For example: ACDEZ.

```
Drives := DriveGetList([Type])
```

## Type

If *Type* is omitted, all drive types are retrieved. Otherwise, *Type* should be one of the following words to retrieve only a specific type of drive: CDROM, REMOVABLE, FIXED, NETWORK, RAMDISK, UNKNOWN.

## DriveGetCapacity

Retrieves the total capacity of *Path* (e.g. C:\) in megabytes.

```
MB := DriveGetCapacity(Path)
```

## Path

Any path contained by the drive.

## DriveGetSpaceFree

Retrieves the free disk space of the drive which contains *Path*, in megabytes (rounded down to the nearest megabyte).

```
MB := DriveGetSpaceFree(Path)
```

## Path

Any path contained by the drive.

## DriveGetFilesystem

Retrieves the type of *Drive's* file system.

```
FS := DriveGetFilesystem(Drive)
```

## Drive

The drive letter followed by a colon and an optional backslash, or a UNC name such `\\server1\share1`.

The return value is one of the following words: FAT, FAT32, NTFS, CDFS (typically indicates a CD), UDF (typically indicates a DVD). If the drive does not contain formatted media, the return value is blank and `ErrorLevel` is set to 1.

## DriveGetLabel

Retrieves *Drive's* volume label.

```
Label := DriveGetLabel(Drive)
```

### Drive

The drive letter followed by a colon and an optional backslash, or a UNC name such `\\server1\share1`.

To change the label, follow this example: `DriveSetLabel("C:", "MyLabel")`.

## DriveGetSerial

Retrieves *Drive's* volume serial number expressed as a decimal integer.

```
Serial := DriveGetSerial(Drive)
```

### Drive

The drive letter followed by a colon and an optional backslash, or a UNC name such `\\server1\share1`.

## DriveGetType

Retrieves *Path's* drive type, which is one of the following words: Unknown, Removable, Fixed, Network, CDROM, RAMDisk.

```
Type := DriveGetType(Path)
```

### Path

Any path contained by the drive.

## DriveGetStatus

Retrieves *Path*'s status.

```
Status := DriveGetStatus(Path)
```

### Path

Any path contained by the drive.

The return value is one of the following strings:

| String   | Notes                                                                                 |
|----------|---------------------------------------------------------------------------------------|
| Unknown  | Might indicate unformatted/RAW file system.                                           |
| Ready    | This is the most common.                                                              |
| NotReady | Typical for removable drives that don't contain media.                                |
| Invalid  | <i>Path</i> does not exist or is a network drive that is presently inaccessible, etc. |

## DriveGetStatusCD

Retrieves the media status of a CD or DVD drive.

```
Status := DriveGetStatusCD([Drive])
```

## Drive

The drive letter followed by a colon. If omitted, the default CD/DVD drive will be used.

The return value is blank if the status cannot be determined. Otherwise, it is one of the following strings:

|           |                                                                                                                                                                                     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| not ready | The drive is not ready to be accessed, perhaps due to being engaged in a write operation. Known limitation: "not ready" also occurs when the drive contains a DVD rather than a CD. |
| open      | The drive contains no disc, or the tray is ejected.                                                                                                                                 |
| playing   | The drive is playing a disc.                                                                                                                                                        |
| paused    | The previously playing audio or video is now paused.                                                                                                                                |
| seeking   | The drive is seeking.                                                                                                                                                               |
| stopped   | The drive contains a CD but is not currently accessing it.                                                                                                                          |

This command will probably not work on a network drive or non-CD/DVD drive; if it fails in such cases or for any other reason, the return value is blank and [ErrorLevel](#) is set to 1.

If the tray was recently closed, there may be a delay before the command completes.

To eject or retract the tray, use [DriveEject](#).

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

Some of the commands will accept a network share name as *Path* or *Drive*, such as `\\MyServer\MyShare\`

In general, *Path* (but not *Drive*) can be any path. Since NTFS supports mounted volumes and directory junctions, different paths with the same drive letter can produce different results (different amounts of free space, capacity, etc.).

## Related

[Drive](#)

## Example

```
; This is a working example script.
folder := DirSelect(, 3, "Pick a drive to
analyze:")
if folder = ""
 return
list := DriveGetList()
cap := DriveGetCapacity(folder)
free := DriveGetSpaceFree(folder)
fs := DriveGetFilesystem(folder)
label := DriveGetLabel(folder)
serial := DriveGetSerial(folder)
type := DriveGetType(folder)
status := DriveGetStatus(folder)
MsgBox("
(Q
```



# DirCopy

Copies a folder along with all its sub-folders and files (similar to xcopy).

```
DirCopy Source, Dest [, Flag]
```

## Parameters

### Source

Name of the source directory (with no trailing backslash), which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.  
For example: `C:\My Folder`

### Dest

Name of the destination directory (with no trailing backslash), which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.  
For example: `C:\Copy of My Folder`

### Flag

(optional) this flag determines whether to overwrite files if they already exist:

**0** (default): Do not overwrite existing files. The operation will fail and have no effect if *Dest* already exists as a file or directory.

**1**: Overwrite existing files. However, any files or subfolders inside *Dest* that do not have a counterpart in *Source* will not be deleted.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise. However, if the source directory contains any saved webpages consisting of a *PageName.htm* file and a corresponding directory named *PageName\_files*, an error may be indicated even when the copy is successful.

## Remarks

If the destination directory structure doesn't exist it will be created if possible.

Since the operation will recursively copy a folder along with all its subfolders and files, the result of copying a folder to a destination somewhere inside itself is undefined. To work around this, first copy it to a destination outside itself, then use `DirMove` to move that copy to the desired location.

`DirCopy` copies a single folder. To instead copy the contents of a folder (all its files and subfolders), see the examples section of `FileCopy`.

## Related

[DirMove](#), [FileCopy](#), [FileMove](#), [FileDelete](#), [file-loops](#), [DirSelect](#), [SplitPath](#)

## Examples

```
DirCopy, C:\My Folder, C:\Copy of My Folder
```

```
; Example #2: A working script that prompts you to
copy a folder.
```

```
DirSelect, SourceFolder, , 3, Select the folder to
copy
if SourceFolder = ""
 return
; Otherwise, continue.
DirSelect, TargetFolder, , 3, Select the folder IN
WHICH to create the duplicate folder.
if TargetFolder = ""
 return
; Otherwise, continue.
Result := MsgBox("A copy of the folder
'%SourceFolder%' will be put into
'%TargetFolder%'. Continue?",, 4)
if Result = "No"
 return
SplitPath, %SourceFolder%, SourceFolderName ;
Extract only the folder name from its full path.
DirCopy, %SourceFolder%,
%TargetFolder%\%SourceFolderName%
if ErrorLevel
 MsgBox("The folder could not be copied,
perhaps because a folder of that name already
exists in '%TargetFolder%'")
return
```

# DirCreate

Creates a directory/folder.

**DirCreate** DirName

**Command Example:** DirCreate A\_ScriptDir  
"\TempDir"

**Function Example:** DirCreate(A\_ScriptDir  
"\TempDir")

## Parameters

### DirName

Name of the directory to create, which is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

## ErrorLevel

ErrorLevel is set to 1 if there was a problem or 0 otherwise.

A\_LastError is set to the result of the operating system's GetLastError() function.

## Remarks

*DirName*

## Related

[DirDelete](#)

## Example

```
DirCreate, C:\Test1\My Images\Folder2
```

# DirDelete

Deletes a folder.

```
DirDelete DirName [, Recurse]
```

```
Command Example: DirDelete A_ScriptDir
"\TempDir", 1
```

```
Function Example: DirDelete(A_ScriptDir
"\TempDir",1)
```

## Parameters

### DirName

Name of the directory to delete, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

### Recurse?

**0** (default): Do **not** remove files and sub-directories contained in *DirName*. In this case, if *DirName* is not empty, no action will be taken and `ErrorLevel` will be set to 1.

**1**: Remove all files and subdirectories (like the Windows command `"rmdir /S"`).

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

None

## Related

`DirCreate`, `FileDelete`

## Example

```
DirDelete, C:\Download Temp
DirDelete, C:\Download Temp, 1
```

# DirMove

Moves a folder along with all its sub-folders and files. It can also rename a folder.

```
DirMove Source, Dest [, Flag]
```

## Example:

```
DirMove("C:\MyFolder", "C:\MyNewFolder")
```

```
Example: DirMove, C:\MyFolder, C:\MyNewFolder
```

## Parameters

### Source

Name of the source directory (with no trailing backslash), which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

For example: C:\My Folder

### Dest

The new path and name of the directory (with no trailing backslash), which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. For example: D:\My Folder. **Note:** *Dest* is the actual path and name that the directory will have after it is moved; it is *not* the directory

into which *Source* is moved (except for the known limitation mentioned below).

## Flag

(options) Specify one of the following single characters:

**0** (default): Do not overwrite existing files. The operation will fail if *Dest* already exists as a file or directory.

**1**: Overwrite existing files. However, any files or subfolders inside *Dest* that do not have a counterpart in *Source* will not be deleted. **Known limitation:** If *Dest* already exists as a folder and it is on the same volume as *Source*, *Source* will be moved into it rather than overwriting it. To avoid this, see the next option.

**2**: The same as mode 1 above except that the limitation is absent.

**R**: Rename the directory rather than moving it. Although renaming normally has the same effect as moving, it is helpful in cases where you want "all or none" behavior; that is, when you don't want the operation to be only partially successful when *Source* or one of its files is locked (in use). Although this method cannot move *Source* onto a different volume, it can move it to any other directory on its own volume. The operation will fail if *Dest* already exists as a file or directory.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

DirMove moves a single folder to a new location. To instead move the contents of a folder (all its files and subfolders), see the examples section of [FileMove](#).

If the source and destination are on different volumes or UNC paths, a copy/delete operation will be performed rather than a move.

## Related

[DirCopy](#), [FileCopy](#), [FileMove](#), [FileDelete](#), [File-loops](#), [DirSelect](#), [SplitPath](#)

## Example

```
DirMove, C:\My Folder, D:\My Folder ; Move to a
new drive.
DirMove, C:\My Folder, C:\My Folder (renamed), R
; Simple rename.
DirMove, C:\My Folder, C:\New Location\My Folder,
R ; Folders can be "renamed into" another
location as long as it's on the same volume.
```

# DirSelect

Displays a standard dialog that allows the user to select a folder.

```
OutputVar := DirSelect(StartingFolder, Options,
Prompt)
```

```
Function example: Folder := DirSelect(A_Temp)
```

## Parameters

### OutputVar

The name of the variable in which to store the user's selected folder. This will be made blank if the user cancels the dialog (i.e. does not wish to select a folder). If the user selects a root directory (such as C:\), *OutputVar* will contain a trailing backslash. If this is undesirable, remove it as follows:

```
DirSelect, Folder
Folder := RegExReplace(Folder, "\\$") ;
Removes the trailing backslash, if present.
```

### StartingFolder

If blank or omitted, the dialog's initial selection will be the user's My Documents folder (or possibly My Computer). A [CLSID folder](#) such as `::{20d04fe0-3aea-1069-a2d8-08002b30309d}` (i.e. My

Computer) may be specified start navigation at a specific special folder.

Otherwise, the most common usage of this parameter is an asterisk followed immediately by the absolute path of the drive or folder to be initially selected. For example, `*C:\` would initially select the C drive. Similarly, `*C:\My Folder` would initially select that particular folder.

The asterisk indicates that the user is permitted to navigate upward (closer to the root) from the starting folder. Without the asterisk, the user would be forced to select a folder inside *StartingFolder* (or *StartingFolder* itself). One benefit of omitting the asterisk is that *StartingFolder* is initially shown in a tree-expanded state, which may save the user from having to click the first plus sign.

If the asterisk is present, upward navigation may optionally be restricted to a folder other than Desktop. This is done by preceding the asterisk with the absolute path of the uppermost folder followed by exactly one space or tab. In the following example, the user would not be allowed to navigate any higher than C:\My Folder (but the initial selection would be C:\My Folder\Projects):

```
C:\My Folder *C:\My Folder\Projects
```

## Options

One of the following numbers:

**0:** The options below are all disabled (except on Windows 2000, where the "make new folder" button might appear anyway).

**1** (default): A button is provided that allows the user to create new folders.

**Add 2** to the above number to provide an edit field that allows the user to type the name of a folder. For example, a value of 3 for this parameter provides both an edit field and a "make new folder" button.

**Add 4** to the above number to omit the `BIF_NEWDIALOGSTYLE` property. Adding 4 ensures that `DirSelect` will work properly even in a Preinstallation Environment like WinPE or BartPE. However, this prevents the appearance of a "make new folder" button, at least on Windows XP.

If the user types an invalid folder name in the edit field, *OutputVar* will be set to the folder selected in the navigation tree rather than what the user entered, at least on Windows XP.

### Prompt

Text displayed in the window to instruct the user what to do. If omitted or blank, it will default to "Select Folder - %A\_SCRIPTNAME%" (i.e. the name of the current script).

### ErrorLevel

`ErrorLevel` is set to 1 if the user dismissed the dialog without selecting a folder (such as by pressing the Cancel button). It is also set to 1 if the system refused to show the dialog (rare). Otherwise, it is set to 0.

## Remarks

A GUI window may display a modal folder-selection dialog by means of the `+OwnDialogs` option. A modal dialog prevents the user from interacting with the GUI window until the dialog is dismissed.

Known limitation: A `timer` that launches during the display of a `DirSelect` dialog will postpone the effect of the user's clicks inside the dialog until after the timer finishes. To work around this, avoid using timers whose subroutines take a long time to finish, or disable all timers during the dialog:

```
Thread, NoTimers
DirSelect, OutputVar,, 3
Thread, NoTimers, false
```

## Related

`FileSelect`, `MsgBox`, `InputBox`, `ToolTip`, `GUI`, `CLSID List`, `DirCopy`, `DirMove`, `SplitPath`

Also, the operating system offers standard dialog boxes that prompt the user to pick a font, color, or icon. These dialogs can be displayed via `DllCall` as demonstrated at [www.autohotkey.com/forum/topic17230.html](http://www.autohotkey.com/forum/topic17230.html).

## Example

```
DirSelect, OutputVar, , 3
if OutputVar = ""
```

```
 MsgBox, You didn't select a folder.
else
 MsgBox, You selected folder "%OutputVar%".
```

```
; CLSID Example:
```

```
DirSelect, OutputVar, ::{20d04fe0-3aea-1069-a2d8-
08002b30309d} ; My Computer.
```

# FileAppend

Writes text or binary data to the end of a file (first creating the file, if necessary).

```
OutputVar := FileAppend([Text, Filename, Encoding])
```

```
Command Example: FileAppend "text",
A_ScriptDir "\MyFile.txt"
Function Example: Success :=
FileAppend("text", A_ScriptDir "\MyFile.txt")
```

## Parameters

### Text

The text to append to the file. This text may include linefeed characters (``n`) to start new lines. In addition, a single long line can be broken up into several shorter ones by means of a [continuation section](#).

*Text* can contain binary data if the **RAW** option is used, but in that case `FileAppend` will always append an even number of bytes (`StrLen` times two). This can be used to [save clipboard data to file](#). `File.RawWrite` can be used to write an odd or even number of bytes.

If *Text* is blank, *Filename* will be created as an empty file (but if the file already exists, its modification time will be updated).

### Filename

The name of the file to be appended, which is assumed to be in `A_WorkingDir` if an absolute path isn't specified.

**Standard Output (stdout):** Specifying an asterisk (\*) for *Filename* causes *Text* to be sent to standard output (stdout). Such text can be redirected to a file, piped to another EXE, or captured by [fancy text editors](#). For example, the following would be valid if typed at a command prompt:

```
"%ProgramFiles%\AutoHotkey\AutoHotkey.exe"
"My Script.ahk" >"Error Log.txt"
```

However, text sent to stdout will not appear at the command prompt it was launched from. This can be worked around by piping a script's output to another command or program. For example:

```
"%ProgramFiles%\AutoHotkey\AutoHotkey.exe"
"My Script.ahk" |more
```

```
For /F "tokens=*" %L in
('" "%ProgramFiles%\AutoHotkey\AutoHotkey.exe"
e" "My Script .ahk"'') do @Echo %L
```

Specifying two asterisks (\*\*) for *Filename* causes *Text* to be sent to the stderr stream.

## Options

Zero or more of the following strings. Separate each option from the next

with a single space or tab. For example: `"`n UTF-8"`

**Encoding:** Specify any of the encoding names accepted by `FileEncoding` (excluding the empty string) to use that encoding if the file lacks a UTF-8 or UTF-16 byte order mark. If omitted, it defaults to `A_FileEncoding`.

**RAW:** Specify the word RAW (case-insensitive) to write the exact bytes contained by *Text* to the file as-is, without any conversion. This option overrides any previously specified encoding and vice versa.

**`n (a linefeed character):** Inserts a carriage return (`^r`) before each linefeed (`^n`) if one is not already present. In other words, it translates from ``n` to `^r^n`. This translation typically does not affect performance. If this option is not used, line endings within *Text* are not changed.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

`A_LastError` is set to the result of the operating system's `GetLastError()` function.

## Remarks

To overwrite an existing file, delete it with `FileDelete` prior to using `FileAppend`.

The target file is automatically closed after the text is appended (except when `FileAppend` is used in its single-parameter mode inside a `file-reading/writing loop`).

`FileOpen()` in append mode provides more control than `FileAppend` and allows the file to be kept open rather than opening and closing it each time. Once a file is opened in append mode, use `file.write(string)` to append the string. File objects also support binary I/O via `RawWrite/RawRead` or `WriteNum/ReadNum`.

## Related

[FileOpen/File Object](#), [FileRead](#), [file-reading loop](#), [IniWrite](#), [FileDelete](#), [OutputDebug](#), [continuation sections](#)

## Example

```
FileAppend "Another line.\n", "C:\My Documents\Test.txt"
```

**; The following example uses a continuation section to enhance readability and maintainability:**

```
FileAppend "
(
A line of text.
By default, the hard carriage return (Enter)
between the previous line and this one will be
written to the file.
 This line is indented with a tab; by
default, that tab will also be written to the
file.
)", A_Desktop "\My File.txt"
```

; The following example demonstrates how to automate FTP uploading using the operating system's built-in FTP command. This script has been tested on Windows XP.

```
FTPCommandFile := A_ScriptDir "\FTPCommands.txt"
FTPLogFile := A_ScriptDir "\FTPLog.txt"
FileDelete FTPCommandFile ; In case previous run
was terminated prematurely.
```

```
FileAppend "
(Q
open host.domain.com
username
password
binary
cd htdocs
put " VarContainingNameOfTargetFile "
delete SomeOtherFile.htm
rename OldFileName.htm NewFileName.htm
ls -l
quit
)", FTPCommandFile
```

```
RunWait A_ComSpec ' /c ftp.exe -s:"'
FTPCommandFile "' >"' FTPLogFile "'
FileDelete FTPCommandFile ; Delete for security
reasons.
Run FTPLogFile ; Display the log for review.
```

# FileCopy

Copies one or more files.

```
FileCopy SourcePattern, DestPattern [, Flag]
```

## Parameters

### SourcePattern

The name of a single file or folder, or a wildcard pattern such as `C:\Temp\*.tmp`. *SourcePattern* is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

### DestPattern

The name or pattern of the destination, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. To perform a simple copy -- retaining the existing file name(s) -- specify only the folder name as shown in these functionally identical examples:

```
FileCopy, C:*.txt, C:\My Folder
```

```
FileCopy, C:*.txt, C:\My Folder*.*
```

### Flag

(optional) this flag determines whether to overwrite files if they already exist:

0 = (default) do not overwrite existing files

1 = overwrite existing files

## ErrorLevel

`ErrorLevel` is set to the number of files that could not be copied due to an error, or 0 otherwise.

In either case, if the source file is a single file (no wildcards) and it does not exist, `ErrorLevel` is set to 0. To detect this condition, use `FileExist` on the source file prior to copying it.

Unlike `FileMove`, copying a file onto itself is always counted as an error, even if the overwrite mode is in effect.

If files were found, `A_LastError` is set to 0 (zero) or the result of the operating system's `GetLastError()` function immediately after the last failure. Otherwise `A_LastError` contains an error code that might indicate why no files were found.

## Remarks

`FileCopy` copies files only. To instead copy the contents of a folder (all its files and subfolders), see the examples section below. To copy a single folder (including its subfolders), use `DirCopy`.

The operation will continue even if error(s) are encountered.

## Related

FileMove, DirCopy, DirMove, FileDelete

## Examples

```
FileCopy, C:\My Documents>List1.txt, D:\Main Backup\ ; Make a copy but keep the orig. file name.
FileCopy, C:\My File.txt, C:\My File New.txt ; Copy a file into the same folder by providing a new name.
FileCopy, C:\Folder1*.txt, D:\New Folder*.bkp ; Copy to new location and give new extension.
```

```
; The following example copies all files and folders inside a folder to a different folder:
ErrorCount := CopyFilesAndFolders("C:\My Folder*.*", "D:\Folder to receive all files & folders")
if ErrorCount <> 0
 MsgBox %ErrorCount% files/folders could not be copied.

CopyFilesAndFolders(SourcePattern, DestinationFolder, DoOverwrite = false)
; Copies all files and folders matching SourcePattern into the folder named DestinationFolder and
; returns the number of files/folders that could not be copied.
{
 ; First copy all the files (but not the folders):
```

```
FileCopy, %SourcePattern%,
%DestinationFolder%, %DoOverwrite%
ErrorCount := ErrorLevel
; Now copy all the folders:
Loop, %SourcePattern%, 2 ; 2 means "retrieve
folders only".
{
 DirCopy, %A_LoopFilePath%,
%DestinationFolder%\%A_LoopFileName%,
%DoOverwrite%
 ErrorCount += ErrorLevel
 if ErrorLevel ; Report each problem
folder by name.
 MsgBox Could not copy %A_LoopFilePath%
into %DestinationFolder%.
}
return ErrorCount
}
```

# FileCreateShortcut

Creates a shortcut (.lnk) file.

```
FileCreateShortcut Target, LinkFile [, WorkingDir,
Args, Description, IconFile, ShortcutKey, IconNumber,
RunState]
```

```
Command Example: FileCreateShortcut
A_ScriptDir "\MyScript.lnk", A_ScriptFullPath
Function Example:
FileCreateShortCut(A_ScriptDir "MyScript.lnk",
A_ScriptFullPath)
```

## Parameters

### Target

Name of the file that the shortcut refers to, which should include an absolute path unless the file is integrated with the system (e.g. Notepad.exe). The file does not have to exist at the time the shortcut is created; in other words, shortcuts to invalid targets can be created.

### LinkFile

Name of the shortcut file to be created, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. Be sure to include the **.lnk** extension. If the file already exists, it will be overwritten.

## WorkingDir

Directory that will become *Target's* current working directory when the shortcut is launched. If blank or omitted, the shortcut will have a blank "Start in" field and the system will provide a default working directory when the shortcut is launched.

## Args

Parameters that will be passed to *Target* when it is launched. Separate parameters with spaces. If a parameter contains spaces, enclose it in double quotes.

## Description

Comments that describe the shortcut (used by the OS to display a tooltip, etc.)

## IconFile

The full path and name of the icon to be displayed for *LinkFile*. It must either be an ico file or the very first icon of an EXE or DLL.

## ShortcutKey

A single letter, number, or the name of a single key from the [key list](#) (mouse buttons and other non-standard keys might not be supported). **Do not** include modifier symbols. Currently, all shortcut keys are created as CTRL+ALT shortcuts. For example, if the letter B is specified for this parameter, the shortcut key will be CTRL-ALT-B.

## IconNumber

To use an icon in *IconFile* other than the first, specify that number here. For example, 2 is the second icon.

### RunState

To have *Target* launched minimized or maximized, specify one of the following digits:

- 1 - Normal (this is the default)
- 3 - Maximized
- 7 - Minimized

### ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

### Remarks

*Target* might not need to include a path if the target file resides in one of the folders listed in the system's PATH environment variable.

The *ShortcutKey* of a newly created shortcut will have no effect unless the shortcut file resides on the desktop or somewhere in the Start Menu. If the *ShortcutKey* you choose is already in use, your new shortcut takes precedence.

An alternative way to create a shortcut to a URL is the following example, which creates a special URL shortcut. Change the first two parameters to suit your preferences:

```
IniWrite, http://www.google.com, C:\My Shortcut.url,
```

```
InternetShortcut, URL.
```

The following may be optionally added to assign an icon to the above:

```
iniwrite, <IconFile>, C:\My Shortcut.url,
InternetShortcut, IconFile
iniwrite, 0, C:\My Shortcut.url,
InternetShortcut, IconIndex
```

In the above, replace 0 with the index of the icon (0 is used for the first icon) and replace <IconFile> with a URL, EXE, DLL, or ICO file. Examples: C:\Icons.dll, C:\App.exe, <http://www.somedomain.com/ShortcutIcon.ico>

The operating system will treat a .URL file created by the above as a real shortcut even though it is a plain text file rather than a .LNK file.

## Related

[FileGetShortcut](#), [FileAppend](#)

## Example

```
; The letter "i" in the last parameter makes the
shortcut key be Ctrl-Alt-I:
```

```
FileCreateShortcut, Notepad.exe, %A_Desktop%\My
Shortcut.lnk, C:\, "%A_ScriptFullPath%", My
Description, C:\My Icon.ico, i
```

# FileDelete

Deletes one or more files.

**FileDelete** FilePattern

```
Command Example: FileDelete A_ScriptDir
"\MyFile.txt"
Function Example: FileDelete(A_ScriptDir
"\MyFile.txt")
```

## Parameters

### FilePattern

The name of a single file or a wildcard pattern such as

`C:\Temp\*.tmp`. *FilePattern* is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

To remove an entire folder, along with all its sub-folders and files, use [DirDelete](#).

## ErrorLevel

`ErrorLevel` is set to the number of files that failed to be deleted (if any) or 0 otherwise. Deleting a wildcard pattern such as `*.tmp` is considered a success even if it does not match any files; thus `ErrorLevel` is set to 0.

If files were found, `A_LastError` is set to 0 (zero) or the result of the operating system's `GetLastError()` function immediately after the last failure. Otherwise `A_LastError` contains an error code that might indicate why no files were found.

## Remarks

To delete a read-only file, first remove the read-only attribute. For example:

```
FileSetAttrib, -R, C:\My File.txt.
```

## Related

[FileRecycle](#), [DirDelete](#), [FileCopy](#), [FileMove](#)

## Example

```
FileDelete, C:\temp files*.tmp
```

# FileEncoding

Sets the default encoding for [FileRead](#), [Loop Read](#), [FileAppend](#), and [FileOpen](#).

**FileEncoding** **[Encoding]**

Encoding can be one of the following values:

- **UTF-8**: Unicode UTF-8, equivalent to CP65001.
- **UTF-16**: Unicode UTF-16 with little endian byte order, equivalent to CP1200.
- **UTF-8-RAW** or **UTF-16-RAW**: As above, but no byte order mark is written when a new file is created.
- **CP $nnn$** : a code page with numeric identifier  $nnn$ . See [Code Page Identifiers](#).
- Empty or omitted: equivalent to **CP0**.

## Remarks

**A\_FileEncoding** contains the current setting.

If not otherwise set by the script, the default is **CP0**.

**CP0** does not universally identify a single code page; rather, it corresponds to the system default ANSI code page, which depends on the system locale or "language for non-Unicode programs" system setting. If **CP0** is the current setting, **A\_FileEncoding** returns **CP0** (never a blank value). To get the

actual code page number, call `DllCall("GetACP")`.

## Related

[FileOpen](#), [StrPut](#), [StrGet](#), [Script Compatibility](#)

# FileGetAttrib

Reports whether a file or folder is read-only, hidden, etc.

```
OutputVar := FileGetAttrib([Filename])
```

```
Function Example: Attr :=
FileGetAttrib(A_AhkPath)
```

```
OutputVar := FileGetAttrib([Filename])
AttributeString := FileExist(FilePattern)
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved text.

### Filename

The name of the target file, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. If omitted, the current file of the innermost enclosing `File-Loop` will be used instead.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

`A_LastError` is set to the result of the operating system's `GetLastError()` function.

## Remarks

The string returned will contain a subset of the letters in the string "RASHNDOCT":

R = READONLY

A = ARCHIVE

S = SYSTEM

H = HIDDEN

N = NORMAL

D = DIRECTORY

O = OFFLINE

C = COMPRESSED

T = TEMPORARY

To check if a particular attribute is present in the retrieved string, following this example:

```
FileGetAttrib, Attributes, C:\My File.txt
if Instr(Attributes, "H")
 MsgBox The file is hidden.
```

On a related note, to retrieve a file's 8.3 short name, follow this example:

```
Loop, Files, C:\My Documents\Address List.txt
 ShortPathName := A_LoopFileShortPath ;
Will yield something similar to
```

```
C:\MYDOCU~1\ADDRES~1.txt
```

A similar method can be used to get the long name of an 8.3 short name.

## Related

[FileExist](#), [FileSetAttrib](#), [FileGetTime](#), [FileSetTime](#), [FileGetSize](#), [FileGetVersion](#),  
[File-loop](#)

## Example

```
FileGetAttrib, OutputVar, C:\New Folder
```

# FileGetShortcut

Retrieves information about a shortcut (.lnk) file, such as its target file.

```
FileGetShortcut LinkFile [, OutTarget, OutDir,
OutArgs, OutDescription, OutIcon, OutIconNum,
OutRunState]
```

```
Command Example: FileGetShortcut A_StartMenu
"\Programs\Internet Explorer.lnk", Target
Function Example: FileGetShortcut(A_StartMenu
"\Programs\Internet Explorer.lnk", Target)
```

## Parameters

### LinkFile

Name of the shortcut file to be analyzed, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. Be sure to include the **.lnk** extension.

### OutTarget (optional)

Name of the variable in which to store the shortcut's target (not including any arguments it might have). For example:

```
C:\WINDOWS\system32\notepad.exe
```

### OutDir (optional)

Name of the variable in which to store the shortcut's working directory.

For example: C:\My Documents. If environment variables such as %WinDir% are present in the string, one way to resolve them is via `StrReplace`. For example: `StrReplace, OutDir, OutDir, `"%WinDir`%", %A_WinDir%`

### OutArgs (optional)

Name of the variable in which to store the shortcut's parameters (blank if none).

### OutDescription (optional)

Name of the variable in which to store the shortcut's comments (blank if none).

### OutIcon (optional)

Name of the variable in which to store the filename of the shortcut's icon (blank if none).

### OutIconNum (optional)

Name of the variable in which to store the shortcut's icon number within the icon file (blank if none). This value is most often 1, which means the first icon.

### OutRunState (optional)

Name of the variable in which to store the shortcut's initial launch state, which is one of the following digits:

1: Normal

3: Maximized

7: Minimized

## ErrorLevel

If there was a problem -- such as *LinkFile* not existing -- all the output variables are made blank and [ErrorLevel](#) is set to 1. Otherwise, [ErrorLevel](#) is set to 0.

## Remarks

Any of the output variables may be omitted if the corresponding information is not needed.

## Related

[FileCreateShortcut](#), [SplitPath](#)

## Example

```
FileSelect, file, 32,, Pick a shortcut to
analyze., Shortcuts (*.lnk)
if file = ""
 return
FileGetShortcut, %file%, OutTarget, OutDir,
OutArgs, OutDesc, OutIcon, OutIconNum, OutRunState
MsgBox
%OutTarget%`n%OutDir%`n%OutArgs%`n%OutDesc%`n%OutI
con%`n%OutIconNum%`n%OutRunState%
```

# FileGetSize

Retrieves the size of a file.

```
OutputVar := FileGetSize([Filename, Units])
```

```
Function Example: FileTime :=
FileGetSize(A_AhkExe)
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved size (rounded down to the nearest whole number).

### Filename

The name of the target file, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. If omitted, the current file of the innermost enclosing `File-Loop` will be used instead.

### Units

If present, this parameter causes the result to be returned in units other than bytes:

K = kilobytes

M = megabytes

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

`A_LastError` is set to the result of the operating system's `GetLastError()` function.

## Remarks

Files of any size are supported, even those over 4 gigabytes, and even if *Units* is bytes.

If the target file is a directory, the size will be reported as whatever the OS believes it to be (probably zero in all cases).

To calculate the size of folder, including all its files, follow this example:

```
FolderSize := 0
DirSelect, WhichFolder ; Ask the user to pick
a folder.
Loop, Files, %WhichFolder%*.*, R
 FolderSize += A_LoopFileSize
MsgBox Size of %WhichFolder% is %FolderSize%
bytes.
```

## Related

[FileGetAttrib](#), [FileSetAttrib](#), [FileGetTime](#), [FileSetTime](#), [FileGetVersion](#), [File-loop](#)

## Example

```
FileGetSize, OutputVar, C:\My Documents\test.doc
; Retrieve the size in bytes.
```

```
FileGetSize, OutputVar, C:\My Documents\test.doc,
K ; Retrieve the size in Kbytes.
```

# FileGetTime

Retrieves the datetime stamp of a file or folder.

```
OutputVar := FileGetTime([Filename, WhichTime])
```

```
Function Example: time :=
FileGetTime(A_AhkPath)
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved date-time in format `YYYYMMDDHH24MISS`. The time is your own local time, not UTC/GMT.

### Filename

The name of the target file or folder, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. If omitted, the current file of the innermost enclosing `File-Loop` will be used instead.

### WhichTime

Which timestamp to retrieve:

M = Modification time (this is the default if the parameter is omitted)

C = Creation time

A = Last access time

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

`A_LastError` is set to the result of the operating system's `GetLastError()` function.

## Remarks

See [YYYYMMDDHH24MISS](#) for an explanation of dates and times.

## Related

[FileSetTime](#), [FormatTime](#), [FileGetAttrib](#), [FileSetAttrib](#), [FileGetSize](#),  
[FileGetVersion](#), [File-loop](#), [DateAdd](#), [DateDiff](#)

## Example

```
FileGetTime, OutputVar, C:\My Documents\test.doc
; Retrieves the modification time by default.
FileGetTime, OutputVar, C:\My Documents\test.doc,
C ; Retrieves the creation time.
```

# FileGetVersion

Retrieves the version of a file.

```
OutputVar := FileGetVersion([Filename])
```

```
Function Example: version :=
FileGetVersion(A_AhkPath)
```

## Parameters

### OutputVar

The name of the variable in which to store the version number/string.

### Filename

The name of the target file. If a full path is not specified, this function uses the search sequence specified by the system [LoadLibrary](#) function. If omitted, the current file of the innermost enclosing [File-Loop](#) will be used instead.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

[A\\_LastError](#) is set to the result of the operating system's [GetLastError\(\)](#) function.

## Remarks

Most non-executable files (and even some EXEs) won't have a version, and thus the OutputVar will be blank in these cases.

## Related

[FileGetAttrib](#), [FileSetAttrib](#), [FileGetTime](#), [FileSetTime](#), [FileGetSize](#), [File-loop](#)

## Example

```
FileGetVersion, version, C:\My Application.exe
FileGetVersion, version,
%A_ProgramFiles%\AutoHotkey\AutoHotkey.exe
```

# FileInstall

Includes the specified file inside the [compiled version](#) of the script.

```
FileInstall Source, Dest [, Flag]
```

```
Command Example: FileInstall "AutoHotkey.dll",
"AutoHotkey.dll"
```

```
Function Example:
FileInstall("AutoHotkey.dll", "AutoHotkey.dll")
```

## Parameters

### Source

The name of the file to be added to the compiled EXE. The file is assumed to be in (or relative to) the script's own directory if an absolute path isn't specified.

The file name **must not** contain double quotes, variable references (e.g. %A\_ProgramFiles%), or wildcards. In addition, any special characters such as literal percent signs and commas must be [escaped](#) (just like in the parameters of all other commands). Finally, this parameter must be listed to the right of the FileInstall command (that is, not on a [continuation line](#) beneath it).

### Dest

When *Source* is extracted from the EXE, this is the name of the file to be

created. It is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. The destination directory must already exist. Unlike *Source*, variable references may be used.

### Flag

(optional) this flag determines whether to overwrite files if they already exist:

0 = (default) do not overwrite existing files

1 = overwrite existing files

### ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

### Remarks

This command is a directive for the [Ahk2Exe compiler](#) that allows you to add extra files to the resulting compiled script. Later, when the compiled script is run, the files are extracted back out onto the disk.

The file is added **during script compilation**. When the compiled script is executed and the same "FileInstall" command is reached, the file is then extracted to *Dest*.

Files added to a script are compressed and also encrypted.

If this command is used in an normal (uncompiled) script, a simple file copy will

be performed instead -- this helps the testing of scripts that will eventually be compiled.

## Related

[FileCopy](#), [#Include](#)

## Example

```
FileInstall, C:\My Documents\My File.txt,
%A_ProgramFiles%\My Application\Readme.txt, 1
```

# FileMove

Moves or renames one or more files.

```
FileMove SourcePattern, DestPattern [, Flag]
```

## Parameters

### SourcePattern

The name of a single file or a wildcard pattern such as C:\Temp\\*.tmp. *SourcePattern* is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

### DestPattern

The name or pattern of the destination, which is assumed to be in %A\_WorkingDir% if an absolute path isn't specified. To perform a simple move -- retaining the existing file name(s) -- specify only the folder name as shown in these functionally identical examples:

```
FileMove, C:*.txt, C:\My Folder
```

```
FileMove, C:*.txt, C:\My Folder*.*
```

### Flag

(optional) this flag determines whether to overwrite files if they already exist:

0 = (default) do not overwrite existing files

1 = overwrite existing files

## ErrorLevel

`ErrorLevel` is set to the number of files that could not be moved due to an error, or 0 otherwise. However, if the source file is a single file (no wildcards) and it does not exist, `ErrorLevel` is set to 0. To detect this condition, use `FileExist` on the source file prior to moving it.

Unlike `FileCopy`, moving a file onto itself is always considered successful, even if the overwrite mode is not in effect.

If files were found, `A_LastError` is set to 0 (zero) or the result of the operating system's `GetLastError()` function immediately after the last failure. Otherwise `A_LastError` contains an error code that might indicate why no files were found.

## Remarks

`FileMove` moves files only. To instead move the contents of a folder (all its files and subfolders), see the examples section below. To move or rename a single folder, use `DirMove`.

The operation will continue even if error(s) are encountered.

Although this command is capable of moving files to a different volume, the operation will take longer than a same-volume move. This is because a same-volume move is similar to a rename, and therefore much faster.

## Related

[FileCopy](#), [DirCopy](#), [DirMove](#), [FileDelete](#)

## Examples

```
FileMove, C:\My Documents>List1.txt, D:\Main
Backup\ ; Move the file without renaming it.
FileMove, C:\File Before.txt, C:\File After.txt ;
Rename a single file.
FileMove, C:\Folder1*.txt, D:\New Folder*.bkp ;
Move and rename files to a new extension.
```

```
; The following example moves all files and
folders inside a folder to a different folder:
ErrorCount := MoveFilesAndFolders("C:\My
Folder*.*", "D:\Folder to receive all files &
folders")
if ErrorCount <> 0
 MsgBox %ErrorCount% files/folders could not be
moved.

MoveFilesAndFolders(SourcePattern,
DestinationFolder, DoOverwrite := false)
; Moves all files and folders matching
SourcePattern into the folder named
DestinationFolder and
; returns the number of files/folders that could
not be moved.
{
 if DoOverwrite = 1
```

```

 DoOverwrite := 2 ; See DirMove for
description of mode 2 vs. 1.
 ; First move all the files (but not the
folders):
 FileMove, %SourcePattern%,
%DestinationFolder%, %DoOverwrite%
 ErrorCount := ErrorLevel
 ; Now move all the folders:
 Loop, Files, %SourcePattern%, D ; D means
"retrieve folders only".
 {
 DirMove, %A_LoopFilePath%,
%DestinationFolder%\%A_LoopFileName%,
%DoOverwrite%
 ErrorCount += ErrorLevel
 if ErrorLevel ; Report each problem
folder by name.
 MsgBox Could not move %A_LoopFilePath%
into %DestinationFolder%.
 }
 return ErrorCount
}

```

# FileOpen

Opens a file.

```
file := FileOpen(Filename, Flags [, Encoding])
```

## Parameters

### Filename

The path of the file to open, which is assumed to be in `A_WorkingDir` if an absolute path isn't specified.

Specify an asterisk (or two) as shown below to open the standard input/output/error stream:

```
FileOpen("*", "r") ; for stdin
FileOpen("*", "w") ; for stdout
FileOpen("**", "w") ; for stderr
```

### Flags

Either a string of characters indicating the desired access mode followed by other options (with optional spaces or tabs in between); or a combination (sum) of numeric flags. Supported values are described in the table below.

### Encoding

The code page to use for text I/O if the file does not contain a UTF-8 or

UTF-16 byte order mark, or if the `h` (handle) flag is used. If omitted, the current value of `A_FileEncoding` is used.

## Flags

| Access modes (mutually-exclusive) |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>r</code>                    | 0     | <i>Read</i> : Fails if the file doesn't exist.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>w</code>                    | 1     | <i>Write</i> : Creates a new file, <b>overwriting any existing file</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>a</code>                    | 2     | <i>Append</i> : Creates a new file if the file didn't exist, otherwise moves the file pointer to the end of the file.                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>rw</code>                   | 3     | <i>Read/Write</i> : Creates a new file if the file didn't exist.                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>h</code>                    |       | Indicates that <i>Filename</i> is a file handle to wrap in an object. Sharing mode flags are ignored and the file or stream represented by the handle is not checked for a byte order mark. The file handle is <b>not</b> closed automatically when the file object is destroyed and calling <code>Close</code> has no effect. Note that <code>Seek</code> , <code>Tell</code> and <code>Length</code> should not be used if <i>Filename</i> is a handle to a nonseeking device such as a pipe or a communications device. |
| Sharing mode flags                |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>-rwd</code>                 |       | Locks the file for read, write and/or delete access. Any combination of <code>r</code> , <code>w</code> and <code>d</code> may be used. Specifying <code>-</code> is the same as specifying <code>-rwd</code> . If omitted entirely, the default is to share all access.                                                                                                                                                                                                                                                   |
|                                   | 0     | If <i>Flags</i> is numeric, the absence of sharing mode flags causes the file to be locked.                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                                   | 0x100 | Shares <i>read</i> access.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|                                   | 0x200 | Shares <i>write</i> access.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|                           |       |                                                                                                                      |
|---------------------------|-------|----------------------------------------------------------------------------------------------------------------------|
|                           | 0x400 | Shares <i>delete</i> access.                                                                                         |
| End of line (EOL) options |       |                                                                                                                      |
| <code>\n</code>           | 4     | Replace <code>\r\n</code> with <code>\n</code> when reading and <code>\n</code> with <code>\r\n</code> when writing. |
| <code>\r</code>           | 8     | Replace standalone <code>\r</code> with <code>\n</code> when reading.                                                |

## Return Value

If the file is opened successfully, the return value is a [File object](#).

If the function fails, the return value is 0 and [A\\_LastError](#) contains an error code.

Use `if file` or `IsObject(file)` to check if the function succeeded.

## Remarks

`File.ReadLine` always supports `\n`, `\r\n` and `\r` as line endings and does not include them in its return value, regardless of whether the `\r` or `\n` options are used. The options only affect translation of line endings within the text returned by `File.Read` or written by `File.Write` or `File.WriteLine`.

When a UTF-8 or UTF-16 file is created, a byte order mark is written to the file **unless** `Encoding` (or `A_FileEncoding` if `Encoding` is omitted) contains "UTF-8-RAW" or "UTF-16-RAW".

When a file containing a UTF-8 or UTF-16 byte order mark (BOM) is opened with read access, the BOM is excluded from the output by positioning the file pointer after it. Therefore, `File.Position` may report 3 or 2 immediately

after opening the file.

## Related

[FileEncoding](#), [File Object](#), [FileRead](#)

## Examples

**; Example: This is a working script that writes some text to a file then reads it back into memory.**

**; It provides the same functionality as this [DllCall-example](#).**

```
FileSelect, FileName, S16,, Create a new file:
```

```
if (FileName = "")
```

```
 return
```

```
file := FileOpen(FileName, "w")
```

```
if !IsObject(file)
```

```
{
```

```
 MsgBox Can't open "%FileName%" for
writing.
```

```
 return
```

```
}
```

```
TestString := "This is a test string.`r`n" ; When
writing a file this way, use `r`n rather than `n
to start a new line.
```

```
file.Write(TestString)
```

```
file.Close()
```

**; Now that the file was written, read its contents back into memory.**

**file := FileOpen(FileName, "r-d") ; read the file  
("r"), share all access except for delete ("-d")**

```
if !IsObject(file)
{
 MsgBox Can't open "%FileName%" for
reading.
 return
}
CharsToRead := StrLen(TestString)
TestString := file.Read(CharsToRead)
file.Close()
MsgBox The following string was read from the
file: %TestString%
```

```
; Open the script in read-only mode and read its
first line:
```

```
file := FileOpen(A_ScriptFullPath, "r")
MsgBox % file.ReadLine()
```

```
; Open a console window for this demonstration:
DllCall("AllocConsole")
```

```
; Open the application's stdin/stdout streams.
```

```
stdin := FileOpen("*", "r")
stdout := FileOpen("*", "w")
stdout.Write("Enter your query.`n\> ")
stdout.Read(0) ; Flush the write buffer.
query := RTrim(stdin.ReadLine(), "`n")
stdout.WriteLine("Your query was '" query "'. Have
a nice day.")
stdout.Read(0) ; Flush the write buffer.
Sleep 5000
```

# FileRead

Reads a file's contents into a [variable](#).

```
OutputVar := FileRead(Filename)
```

```
Function Example: file := FileRead(A_ScriptDir
"\MyFile.txt")
```

## Parameters

### Filename

The name of the file to read, which is assumed to be in [A\\_WorkingDir](#) if an absolute path isn't specified.

### Options

Zero or more of the following strings. Separate each option from the next with a single space or tab. For example: `" n m5000 UTF-8"`

**Encoding:** Specify any of the encoding names accepted by [FileEncoding](#) (excluding the empty string) to use that encoding if the file lacks a UTF-8 or UTF-16 byte order mark. If omitted, it defaults to [A\\_FileEncoding](#).

**RAW:** Specify the word RAW (case-insensitive) to read the file's content as [raw binary data](#). This option overrides any previously specified encoding and vice versa.

**m1024:** If this option is omitted, the entire file is loaded unless there is insufficient memory, in which case an error message is shown and the thread exits (but [Try](#) can be used to avoid this). Otherwise, replace 1024 with a decimal or hexadecimal number of bytes. If the file is larger than this, only its leading part is loaded. Note: This might result in the last line ending in a naked carriage return (^r) rather than ^r^n.

**^n** (a linefeed character): Replaces any/all occurrences of carriage return & linefeed (^r^n) with linefeed (^n). However, this translation reduces performance and is usually not necessary. For example, text containing ^r^n is already in the right format to be added to a [Gui Edit control](#). The following [parsing loop](#) will work correctly regardless of whether each line ends in ^r^n or just ^n: `Loop Parse, MyFileContents, ^n, ^r`.

## ErrorLevel

[ErrorLevel](#) is set to 0 if the load was successful. It is set to 1 if a problem occurred such as: 1) file does not exist; 2) file is locked or inaccessible; 3) the system lacks sufficient memory to load the file.

[A\\_LastError](#) is set to the result of the operating system's `GetLastError()` function.

## Reading Binary Data

When the `RAW` option is used, the return value is a string containing the raw, unmodified contents of the file. As the [native encoding](#) is UTF-16, the string

always contains an even number of bytes. To get the number bytes which were read (rounded up to an even number), multiply the return value of [StrLen](#) by 2.

This option is generally required for reading binary data because by default, any bytes read from file are interpreted as text and may be converted from the source file's encoding (as specified in the options or by [A\\_FileEncoding](#)) to the script's [native encoding](#), UTF-16. If the data is not UTF-16 text, this conversion generally changes the data in undesired ways.

If the data contains UTF-16 text, note that many functions assume the data ends at the first binary zero (if any are present). For example, `MsgBox` will only show characters up to the first binary zero. However, the entire contents are still present and can be copied between variables, passed to or returned from functions, concatenated or compared with other binary strings, or be accessed by advanced methods such as [NumGet\(\)](#).

For a demonstration of the RAW option, see [Saving and Restoring the Clipboard](#).

Finally, [FileOpen](#) and [File.RawRead](#) or [File.ReadNum](#) may be used to read binary data without first reading the entire file into memory.

## Remarks

When the goal is to load all or a large part of a file into memory, `FileRead` performs much better than using a [file-reading loop](#).

A file greater than 4 GB in size will cause an exception to be thrown unless the `*m` option is present, in which case the leading part of the file is loaded. An

exception will also be thrown if the program is unable to allocate enough memory to contain the requested amount of data.

If there is concern about using too much memory, check the file size beforehand with `FileGetSize`.

`FileOpen` provides more advanced functionality than `FileRead`, such as reading or writing data at a specific location in the file without reading the entire file into memory. See [File Object](#) for a list of functions.

## Related

[FileEncoding](#), [FileOpen/File Object](#), [file-reading loop](#), [FileGetSize](#), [FileAppend](#), [IniRead](#), [Sort](#), [Download](#)

## Examples

```
; Example #1: Read text file into OutputVar.
MyText := FileRead("C:\My Documents\My File.txt")
```

```
; Example #2: Quickly sort the contents of a file.
Contents := FileRead("C:\Address List.txt")
if not ErrorLevel ; Successfully loaded.
{
 Contents := Sort(Contents)
 FileDelete "C:\Address List
(alphabetical).txt"
 FileAppend Contents, "C:\Address List
(alphabetical).txt"
 Contents := "" ; Free the memory.
}
```



# FileRecycle

Sends a file or directory to the recycle bin, if possible.

**FileRecycle** FilePattern

```
Command Example: FileRecycle A_ScriptDir
"\TempFile.txt"
Function Example: FileRecycle(A_ScriptDir
"\TempFile.txt")
```

## Parameters

### FilePattern

The name of a single file or a wildcard pattern such as C:\Temp\\*.tmp. *FilePattern* is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

To recycle an entire directory, provide its name without a trailing backslash.

## ErrorLevel

**ErrorLevel** is set to 1 if there was a problem or 0 otherwise.

## Remarks

**SHFileOperation** is used to do the actual work. This function may permanently

delete the file if it is too large to be recycled; also, a warning should be shown before this occurs.

## Related

[FileRecycleEmpty](#), [FileDelete](#), [FileCopy](#), [FileMove](#)

## Example

```
FileRecycle, C:\temp files*.tmp
```

# FileRecycleEmpty

Empties the recycle bin.

```
FileRecycleEmpty [, DriveLetter]
```

```
Command Example: FileRecycleEmpty "C:\"
Function Example: FileRecycleEmpty("C:\")
```

## Parameters

### DriveLetter

If omitted, the recycle bin for all drives is emptied. Otherwise, specify a drive letter such as C:\

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

This command requires that MS Internet Explorer 4 or later be installed.

## Related

[FileRecycle](#), [FileDelete](#), [FileCopy](#), [FileMove](#)

## Example

```
FileRecycleEmpty, C:\
```

# FileReplace

Deletes the file and writes text to the file after creating it.

```
OutputVar := FileReplace(Text, Filename [, Encoding])
```

```
Command Example: FileReplace
"text",A_ScriptDir "\MyFile.txt"
Function Example: Success :=
FileReplace("text",A_ScriptDir "\MyFile.txt")
```

## Parameters

### OutputVar

The name of the [variable](#) in which to store true if file was replaced or false otherwise.

### Text

The text to append to the file. This text may include linefeed characters (`\n`) to start new lines. In addition, a single long line can be broken up into several shorter ones by means of a [continuation section](#).

If *Text* is blank, *Filename* will be created as an empty file (but if the file already exists, its modification time will be updated).

If *Text* is `%ClipboardAll%` or a variable that was previously assigned the value of `ClipboardAll`, *Filename* will be unconditionally overwritten with

the entire contents of the clipboard (i.e. `FileDelete` is not necessary).

### Filename

The name of the file to be appended, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

**Binary mode:** To append in binary mode rather than text mode, prepend an asterisk to the filename. This causes each linefeed character (`\n`) to be written as as a single linefeed (LF) rather than the Windows standard of CR+LF. For example: `*C:\My Unix File.txt`.

### Encoding

Overrides the default encoding set by `FileEncoding`, where *Encoding* follows the same format.

### ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

`A_LastError` is set to the result of the operating system's `GetLastError()` function.

### Remarks

The target file is automatically closed after the text is appended (except when `FileAppend` is used in its single-parameter mode inside a [file-reading/writing loop](#)).

`FileOpen` in append mode provides more control than `FileAppend` and allows the

file to be kept open rather than opening and closing it each time. Once a file is opened in append mode, use `file.write(string)` to append the string. File objects also support binary I/O via `RawWrite/RawRead` or `WriteNum/ReadNum`, whereas `FileAppend` supports only text.

## Related

[FileOpen/File Object](#), [FileRead](#), [FileAppend](#), [file-reading loop](#), [IniWrite](#), [FileDelete](#), [OutputDebug](#), [continuation sections](#)

## Example

```
FileAppend, Another line.`n, C:\My
Documents\Test.txt
```

**; The following example uses a continuation section to enhance readability and maintainability:**

```
FileAppend,
(
A line of text.
By default, the hard carriage return (Enter)
between the previous line and this one will be
written to the file.
 This line is indented with a tab; by default,
that tab will also be written to the file.
Variable references such as %Var% are expanded by
default.
), C:\My File.txt
```



# FileSelect

Displays a standard dialog that allows the user to open or save file(s).

```
OutputVar := FileSelect([Options, RootDir\Filename,
Prompt, Filter])
```

```
Function Example: Files := FileSelect("M3",
A_ScriptDir)
```

## Parameters

### OutputVar

The name of the variable in which to store the filename(s) selected by the user. This will be made blank if the user cancels the dialog (i.e. does not wish to select a file).

### Options

If omitted, it will default to zero, which is the same as having none of the options below.

**M**: Multi-select. Specify the letter M to allow the user to select more than one file via shift-click, control-click, or other means. **M** may optionally be followed by a number as described below (for example, both M and M1 are valid). To extract the individual files, see the example at the bottom of this page.

**S:** Save button. Specify the letter S to cause the dialog to always contain a Save button instead of an Open button. S may optionally be followed by a number (or sum of numbers) as described below (for example, both S and S24 are valid).

Even if **M** and **S** are absent, the following numbers can be used. To put more than one of them into effect, add them up. For example, to use 8 and 16, specify the number 24.

**1:** File Must Exist

**2:** Path Must Exist

**8:** Prompt to Create New File

**16:** Prompt to OverWrite File

**32:** Shortcuts (.lnk files) are selected as-is rather than being resolved to their targets. This option also prevents navigation into a folder via a folder shortcut.

If the "Prompt to Overwrite" option is present without the "Prompt to Create" option, the dialog will contain a Save button rather than an Open button. This behavior is due to a quirk in Windows.

### RootDir\Filename

If present, this parameter contains one or both of the following:

**RootDir:** The root (starting) directory, which is assumed to be a subfolder in %A\_WorkingDir% if an absolute path is not specified. If omitted or blank, the starting directory will be a default that might depend on the OS

version (it will likely be the directory most recently selected by the user during a prior use of FileSelect). On Windows XP/2003 and earlier, a **CLSID** such as `::{20d04fe0-3aea-1069-a2d8-08002b30309d}` (i.e. My Computer) may also be specified, in which case any subdirectory present after the CLSID should end in a backslash (otherwise, the string after the last backslash will be interpreted as the default filename, below).

**Filename:** The default filename to initially show in the dialog's edit field. Only the naked filename (with no path) will be shown. To ensure that the dialog is properly shown, ensure that no illegal characters are present (such as `/<|:"`).

Examples:

```
C:\My Pictures\Default Image Name.gif ;
Both RootDir and Filename are present.
C:\My Pictures ; Only RootDir is present.
My Pictures ; Only RootDir is present, and
it's relative to the current working
directory.
My File ; Only Filename is present (but if
"My File" exists as a folder, it is assumed
to be RootDir).
```

### Prompt

Text displayed in the window to instruct the user what to do. If omitted or blank, it will default to "Select File - %A\_SCRIPTNAME%" (i.e. the name of the current script).

## Filter

Indicates which types of files are shown by the dialog.

Example: Documents (\*.txt)

Example: Audio (\*.wav; \*.mp2; \*.mp3)

If omitted, the filter defaults to All Files (\*.\*). An option for Text Documents (\*.txt) will also be available in the dialog's "files of type" menu.

Otherwise, the filter uses the indicated string but also provides an option for All Files (\*.\*) in the dialog's "files of type" drop-down list. To include more than one file extension in the filter, separate them with semicolons as illustrated in the example above.

## ErrorLevel

`ErrorLevel` is set to 1 if the user dismissed the dialog without selecting a file (such as by pressing the Cancel button). It is also set to 1 if the system refused to show the dialog (rare). Otherwise, it is set to 0.

## Remarks

If the user didn't select anything (e.g. pressed CANCEL), *OutputVar* is made blank.

If multi-select is not in effect, *OutputVar* is set to the full path and name of the single file chosen by the user.

If the M option (multi-select) is in effect, *OutputVar* is set to a list of items, each of which except the last is followed by a linefeed (`\n`) character. The first item in the list is the path that contains all the selected files (this path will end in a backslash only if it is a root folder such as `C:\`). The other items are the selected filenames (without path). For example:

```
C:\My Documents\New Folder [this is the path in
which all the files below reside]
test1.txt [these are the naked filenames: no
path info]
test2.txt
... etc.
```

(The example at the bottom of this page demonstrates how to extract the files one by one.)

When multi-select is in effect, the sum of the lengths of the selected filenames is limited to 64 KB. Although this is typically enough to hold several thousand files, *OutputVar* will be made blank if the limit is exceeded.

A GUI window may display a modal file-selection dialog by means of the `+OwnDialogs` option. A modal dialog prevents the user from interacting with the GUI window until the dialog is dismissed.

Known limitation: A `timer` that launches during the display of a `FileSelect` dialog will postpone the effect of the user's clicks inside the dialog until after the timer finishes. To work around this, avoid using timers whose subroutines take a long time to finish, or disable all timers during the dialog:

```
Thread, NoTimers
FileSelect, OutputVar
Thread, NoTimers, false
```

## Related

[DirSelect](#), [MsgBox](#), [InputBox](#), [ToolTip](#), [GUI](#), [CLSID List](#), [parsing loop](#), [SplitPath](#)

Also, the operating system offers standard dialog boxes that prompt the user to pick a font, color, or icon. These dialogs can be displayed via [DllCall](#) as demonstrated at [www.autohotkey.com/forum/topic17230.html](http://www.autohotkey.com/forum/topic17230.html).

## Examples

```
FileSelect, SelectedFile, 3, , Open a file, Text
Documents (*.txt; *.doc)
if SelectedFile = ""
 MsgBox, The user didn't select anything.
else
 MsgBox, The user selected the
following: `n%SelectedFile%
```

```
; CLSID Example (requires XP/2003 or earlier):
FileSelect, OutputVar,, ::{645ff040-5081-101b-
9f08-00aa002f954e} ; Recycle Bin.
```

```
; Multi-Select Example:
FileSelect, files, M3 ; M3 = Multiselect existing
files.
if files = ""
```



# FileSetAttrib

Changes the attributes of one or more files or folders. Wildcards are supported.

```
FileSetAttrib Attributes [, FilePattern,
OperateOnFolders?, Recurse?]
```

```
Command Example: FileSetAttrib "+N",
A_ScriptFullPath
```

```
Function Example: FileSetAttrib("+N",
A_ScriptFullPath)
```

## Parameters

### Attributes

The attributes to change (see Remarks).

### FilePattern

The name of a single file or folder, or a wildcard pattern such as `C:\Temp\*.tmp`. *FilePattern* is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

If omitted, the current file of the innermost enclosing [File-Loop](#) will be used instead.

### Mode

Zero or more of the following letters:

**D**: Include directories (folders).

**F**: Include files. If both F and D are omitted, files are included but not folders.

**R**: Subfolders are recursed into so that files and folders contained therein are operated upon if they match *FilePattern*. All subfolders will be recursed into, not just those whose names match *FilePattern*. If R is omitted, files and folders in subfolders are not included.

Note: If *FilePattern* is a single folder rather than a wildcard pattern, it will always be operated upon regardless of this setting.

## ErrorLevel

**ErrorLevel** is set to the number of files that failed to be changed or 0 otherwise.

If files were found, **A\_LastError** is set to 0 (zero) or the result of the operating system's GetLastError() function immediately after the last failure. Otherwise **A\_LastError** contains an error code that might indicate why no files were found.

## Remarks

**Known limitation:** Files and folders with a complete path name longer than 259 characters are skipped over, and may or may not be counted in **ErrorLevel**. Such files are rare because normally, the operating system does not allow their creation.

The *Attributes* parameter consists of a collection of operators and attribute letters.

## Operators:

|   |                                                                       |
|---|-----------------------------------------------------------------------|
| + | Turn on the attribute                                                 |
| - | Turn off the attribute                                                |
| ^ | Toggle the attribute (set it to the opposite value of what it is now) |

## Attribute letters:

R = READONLY

A = ARCHIVE

S = SYSTEM

H = HIDDEN

N = NORMAL (this is valid only when used without any other attributes)

O = OFFLINE

T = TEMPORARY

Note: Currently, the compression state of files cannot be changed with this command.

## Related

[FileGetAttrib](#), [FileGetTime](#), [FileSetTime](#), [FileGetSize](#), [FileGetVersion](#), [File-loop](#)

## Examples

```
FileSetAttrib, +RH, C:\MyFiles*.*, DF ; +RH is
identical to +R+H
```

```
FileSetAttrib, ^H, C:\MyFiles ; Toggle the
folder's "hidden" attribute.
FileSetAttrib, -R+A, C:\New Text File.txt
FileSetAttrib, +A, C:*.ini, R ; Recurse through
all .ini files on the C drive.
```

# FileSetTime

Changes the datetime stamp of one or more files or folders. Wildcards are supported.

```
FileSetTime [YYYYMMDDHH24MISS, FilePattern,
WhichTime, OperateOnFolders?, Recurse?]
```

```
Command Example: FileSetTime A_Now,
A_ScriptFullPath
Function Example: FileSetTime(A_Now,
A_ScriptFullPath)
```

## Parameters

### YYYYMMDDHH24MISS

If blank or omitted, it defaults to the current time. Otherwise, specify the time to use for the operation (see Remarks for the format). Years prior to 1601 are not supported.

### FilePattern

The name of a single file or folder, or a wildcard pattern such as C:\Temp\\*.tmp. *FilePattern* is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

If omitted, the current file of the innermost enclosing [File-Loop](#) will be used instead.

## WhichTime

Which timestamp to set:

M = Modification time (this is the default if the parameter is blank or omitted)

C = Creation time

A = Last access time

## Mode

Zero or more of the following letters:

**D**: Include directories (folders).

**F**: Include files. If both F and D are omitted, files are included but not folders.

**R**: Subfolders are recursed into so that files and folders contained therein are operated upon if they match *FilePattern*. All subfolders will be recursed into, not just those whose names match *FilePattern*. If R is omitted, files and folders in subfolders are not included.

Note: If *FilePattern* is a single folder rather than a wildcard pattern, it will always be operated upon regardless of this setting.

## ErrorLevel

**ErrorLevel** is set to the number of files that failed to be changed or 0 otherwise. If the specified timestamp is invalid, or *FilePattern* resolves to a blank value, ErrorLevel is set to 1.

If files were found, `A_LastError` is set to 0 (zero) or the result of the operating system's `GetLastError()` function immediately after the last failure. Otherwise `A_LastError` contains an error code that might indicate why no files were found.

## Remarks

**Known limitation:** Files and folders with a complete path name longer than 259 characters are skipped over, and may or may not be counted in `ErrorLevel`. Such files are rare because normally, the operating system does not allow their creation.

A file's last access time might not be as precise on FAT16 & FAT32 volumes as it is on NTFS volumes.

The elements of the `YYYYMMDDHH24MISS` format are:

|      |                                                                                   |
|------|-----------------------------------------------------------------------------------|
| YYYY | The 4-digit year                                                                  |
| MM   | The 2-digit month (01-12)                                                         |
| DD   | The 2-digit day of the month (01-31)                                              |
| HH24 | The 2-digit hour in 24-hour format (00-23). For example, 09 is 9am and 21 is 9pm. |
| MI   | The 2-digit minutes (00-59)                                                       |
| SS   | The 2-digit seconds (00-59)                                                       |

If only a partial string is given for `YYYYMMDDHH24MISS` (e.g. 200403), any remaining element that has been omitted will be supplied with the following default values:

MM: Month 01

DD: Day 01

HH24: Hour 00

MI: Minute 00

SS: Second 00

The built-in variable `A_Now` contains the current local time in the above format. Similarly, `A_NowUTC` contains the current Coordinated Universal Time.

**Note:** Date-time values can be compared, added to, or subtracted from via `DateAdd` and `DateDiff`. Also, it is best to not use greater-than or less-than to compare times unless they are both the same string length. This is because they would be compared as numbers; for example, 20040201 is always numerically less (but chronologically greater) than 200401010533. So instead use `DateDiff` to find out whether the amount of time between them is positive or negative.

## Related

[FileGetTime](#), [FileGetAttrib](#), [FileSetAttrib](#), [FileGetSize](#), [FileGetVersion](#), [FormatTime](#), [File-loop](#), [DateAdd](#), [DateDiff](#)

## Example

```
; Set the modification time to the current time
for all matching files:
FileSetTime, , C:\temp*.txt
```

```
; Set the modification date (time will be
midnight):
```

```
FileSetTime, 20040122, C:\My Documents\test.doc
```

```
; Set the creation date. The time will be set to
4:55pm:
```

```
FileSetTime, 200401221655, C:\My
Documents\test.doc, C
```

```
; Change the mod-date of all files that match a
pattern.
```

```
; Any matching folders will also be changed due to
the last parameter:
```

```
FileSetTime, 20040122165500, C:\Temp*.*, M, DF
```

# IniDelete

Deletes a value from a standard format .ini file.

```
IniDelete Filename, Section [, Key]
```

```
Command Example: IniDelete A_ScriptDir
"\MyIni.ini", "Settings", "AlwaysOnTop"
Function Example: IniDelete(A_ScriptDir
"\MyIni.ini", "Settings", "AlwaysOnTop")
```

## Parameters

### Filename

The name of the .ini file, which is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

### Section

The section name in the .ini file, which is the heading phrase that appears in square brackets (do not include the brackets in this parameter).

### Key

The key name in the .ini file. **If omitted, the entire *Section* will be deleted.**

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

A standard ini file looks like:

```
[SectionName]
Key=Value
```

## Related

[IniRead](#), [IniWrite](#), [RegDelete](#)

## Example

```
IniDelete, C:\Temp\myfile.ini, section2, key
```

# IniRead

Reads a value, section or list of section names from a standard format .ini file.

```
OutputVar := IniRead(Filename, Section, Key [, Default])
OutputVarSection := IniRead(Filename, Section)
OutputVarSectionNames := IniRead(Filename)
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved value. If the value cannot be retrieved, the variable is set to the value indicated by the *Default* parameter (described below).

### OutputVarSection

Omit the *Key* parameter to read an entire section. Comments and empty lines are omitted. Only the first 65,533 characters of the section are retrieved.

### OutputVarSectionNames

Omit the *Key* and *Section* parameters to retrieve a linefeed (`\n`) delimited list of section names.

### Filename

The name of the .ini file, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

### Section

The section name in the .ini file, which is the heading phrase that appears in square brackets (do not include the brackets in this parameter).

### Key

The key name in the .ini file.

### Default

The value to store in *OutputVar* if the requested key is not found. If omitted, it defaults to an empty string.

This parameter is not used if *Key* is omitted.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

The operating system automatically omits leading and trailing spaces/tabs from the retrieved string. To prevent this, enclose the string in single or double quote marks. The outermost set of single or double quote marks is also omitted, but any spaces inside the quote marks are preserved.

Values longer than 65,535 characters are likely to yield inconsistent results.

A standard ini file looks like:

```
[SectionName]
Key=Value
```

**Unicode:** IniRead and IniWrite rely on the external functions `GetPrivateProfileString` and `WritePrivateProfileString` to read and write values. These functions support Unicode only in UTF-16 files; all other files are assumed to use the system's default ANSI code page.

## Related

[IniDelete](#), [IniWrite](#), [RegRead](#), [file-reading loop](#), [FileRead](#)

## Example

```
IniRead, OutputVar, C:\Temp\myfile.ini, section2,
key
MsgBox, The value is %OutputVar%.
```

# IniWrite

Writes a value or section to a standard format .ini file.

```
IniWrite Value, Filename, Section, Key
```

```
IniWrite Pairs, Filename, Section
```

## Parameters

### Value

The string or number that will be written to the right of *Key*'s equal sign (=).

If the text is long, it can be broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

### Pairs

The complete content of a section to write to the .ini file, excluding the [SectionName] header. *Key* must be omitted. *Pairs* must not contain any blank lines. If the section already exists, everything up to the last key=value pair is overwritten. *Pairs* can contain lines without an equal sign (=), but this may produce inconsistent results. Comments can be written to the file but are stripped out when they are read back by [IniRead](#).

## Filename

The name of the .ini file, which is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

## Section

The section name in the .ini file, which is the heading phrase that appears in square brackets (do not include the brackets in this parameter).

## Key

The key name in the .ini file.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

Values longer than 65,535 characters can be written to the file, but may produce inconsistent results as they usually cannot be read correctly by `IniRead` or other applications.

A standard ini file looks like:

```
[SectionName]
Key=Value
```

New files are created in either the system's default ANSI code page or UTF-16, depending on [the version of AutoHotkey](#). UTF-16 files may appear to begin with

a blank line, as the first line contains the UTF-16 byte order mark. See below for a workaround.

**Unicode:** IniRead and IniWrite rely on the external functions

[GetPrivateProfileString](#) and [WritePrivateProfileString](#) to read and write values.

These functions support Unicode only in UTF-16 files; all other files are assumed to use the system's default ANSI code page. In [Unicode scripts](#), IniWrite uses UTF-16 for each new file. If this is undesired, ensure the file exists before calling IniWrite. For example:

```
FileAppend,, NonUnicode.ini, CP0 ; The last
parameter is optional in most cases.
```

## Related

[IniDelete](#), [IniRead](#), [RegWrite](#)

## Example

```
IniWrite, this is a new value, C:\Temp\myfile.ini,
section2, key
```

# Loop (files & folders)

Retrieves the specified files or folders, one at a time.

```
Loop Files, FilePattern [, Mode]
```

## Parameters

### Files

The literal word `Files` (case-insensitive). This cannot be a variable or expression.

### FilePattern

The name of a single file or folder, or a wildcard pattern such as `C:\Temp\*.tmp`. *FilePattern* is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

Both asterisks and question marks are supported as wildcards. A match occurs when the pattern appears in either the file's long/normal name or its [8.3 short name](#).

If this parameter is a single file or folder (i.e. no wildcards) and *Recurse* is set to 1 or *Mode* includes `R`, more than one match will be found if the specified file name appears in more than one of the folders being searched.

### Mode

Zero or more of the following letters:

**D**: Include directories (folders).

**F**: Include files. If both F and D are omitted, files are included but not folders.

**R**: Recurse into subdirectories (subfolders). All subfolders will be recursed into, not just those whose names match *FilePattern*. If R is omitted, files and folders in subfolders are not included.

## Special Variables Available Inside a File-Loop

The following variables exist within any file-loop. If an inner file-loop is enclosed by an outer file-loop, the innermost loop's file will take precedence:

|                    |                                                                                                                                                                                                                                                                                                                                                  |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_LoopFileName     | The name of the file or folder currently retrieved (without the path).                                                                                                                                                                                                                                                                           |
| A_LoopFileExt      | The file's extension (e.g. TXT, DOC, or EXE). The period (.) is not included.                                                                                                                                                                                                                                                                    |
| A_LoopFilePath     | The path and name of the file/folder currently retrieved. If <i>FilePattern</i> contains a relative path rather than an absolute path, the path here will also be relative. In addition, any short (8.3) folder names in <i>FilePattern</i> will still be short (see next item to get the long version).                                         |
| A_LoopFileFullPath | This is different than A_LoopFilePath in the following ways: 1) It always contains the absolute/complete path of the file even if <i>FilePattern</i> contains a relative path; 2) Any short (8.3) folder names in <i>FilePattern</i> itself are converted to their long names; 3) Characters in <i>FilePattern</i> are converted to uppercase or |

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            | <p>lowercase to match the case stored in the file system. This is useful for converting file names -- such as those passed into a script as command line parameters -- to their exact path names as shown by Explorer.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <p>A_LoopFileShortPath</p> | <p>The 8.3 short path and name of the file/folder currently retrieved. For example:<br/> C:\MYDOCU~1\ADDRES~1.txt. If <i>FilePattern</i> contains a relative path rather than an absolute path, the path here will also be relative.</p> <p>To retrieve the complete 8.3 path and name for a single file or folder, specify its name for <i>FilePattern</i> as in this example:</p> <pre data-bbox="721 898 1351 1138"> Loop, Files, C:\My Documents\Address List.txt     ShortPathName :=     A_LoopFileShortPath </pre> <p>NOTE: This variable will be <b>blank</b> if the file does not have a short name, which can happen on systems where <code>NtfsDisable8dot3NameCreation</code> has been set in the registry. It will also be blank if <i>FilePattern</i> contains a relative path and the body of the loop uses <a href="#">SetWorkingDir</a> to switch away from the working directory in effect for the loop itself.</p> |
| <p>A_LoopFileShortName</p> | <p>The 8.3 short name, or alternate name of the file. If the file doesn't have one (due to the long name being shorter than 8.3 or perhaps because short-name generation is disabled on an NTFS file system), <i>A_LoopFileName</i> will be retrieved instead.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|                        |                                                                                                                                                                                                                                                             |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_LoopFileDir          | The path of the directory in which <i>A_LoopFileName</i> resides. If <i>FilePattern</i> contains a relative path rather than an absolute path, the path here will also be relative. A root directory will not contain a trailing backslash. For example: C: |
| A_LoopFileTimeModified | The time the file was last modified. Format <code>YYYYMMDDHH24MISS</code> .                                                                                                                                                                                 |
| A_LoopFileTimeCreated  | The time the file was created. Format <code>YYYYMMDDHH24MISS</code> .                                                                                                                                                                                       |
| A_LoopFileTimeAccessed | The time the file was last accessed. Format <code>YYYYMMDDHH24MISS</code> .                                                                                                                                                                                 |
| A_LoopFileAttrib       | The <b>attributes</b> of the file currently retrieved.                                                                                                                                                                                                      |
| A_LoopFileSize         | The size in bytes of the file currently retrieved. Files larger than 4 gigabytes are also supported.                                                                                                                                                        |
| A_LoopFileSizeKB       | The size in Kbytes of the file currently retrieved, rounded down to the nearest integer.                                                                                                                                                                    |
| A_LoopFileSizeMB       | The size in Mbytes of the file currently retrieved, rounded down to the nearest integer.                                                                                                                                                                    |

## Remarks

A file-loop is useful when you want to operate on a collection of files and/or folders, one at a time.

All matching files are retrieved, including hidden files. By contrast, OS features such as the DIR command omit hidden files by default. To avoid processing hidden, system, and/or read-only files, use something like the following inside the loop:

```
if A_LoopFileAttrib contains H,R,S ; Skip any
file that is either H (Hidden), R (Read-only),
or S (System). Note: No spaces in "H,R,S".
 continue ; Skip this file and move on to
the next one.
```

To retrieve files' relative paths instead of absolute paths during a recursive search, use [SetWorkingDir](#) to change to the base folder prior to the loop, and then omit the path from the Loop (e.g. `Loop, Files, *.* , R`). That will cause `A_LoopFilePath` to contain the file's path relative to the base folder.

A file-loop can disrupt itself if it creates or renames files or folders within its own purview. For example, if it renames files via [FileMove](#) or other means, each such file might be found twice: once as its old name and again as its new name. To work around this, rename the files only after creating a list of them. For example:

```
FileList := ""
Loop, Files, *.jpg
 FileList .= A_LoopFileName "`n"
Loop, Parse, %FileList%, `n
 FileMove, %A_LoopField%,
renamed_%A_LoopField%
```

Files in an NTFS file system are probably always retrieved in alphabetical order. Files in other file systems are retrieved in no particular order. To ensure a particular ordering, use the [Sort](#) command as shown in the Examples section below.

Files and folders with a complete path name longer than 259 characters are skipped over as though they do not exist. Such files are rare because normally, the operating system does not allow their creation.

See [Loop](#) for information about [Blocks](#), [Break](#), [Continue](#), and the `A_Index` variable (which exists in every type of loop).

## Related

[Loop](#), [Break](#), [Continue](#), [Blocks](#), [SplitPath](#), [FileSetAttrib](#), [FileSetTime](#)

## Examples

```
; Example #1:
Loop Files, %A_ProgramFiles%*.txt, R ; Recurse
into subfolders.
{
 Result := MsgBox("Filename =
%A_LoopFilePath%\n\nContinue?", , 4)
 if Result = "No"
 break
}
```

```
; Example #2: Calculate the size of a folder,
including the files in all its subfolders:
FolderSizeKB := 0
DirSelect, WhichFolder ; Ask the user to pick a
folder.
Loop, Files, %WhichFolder%*.*, R
```

```
FolderSizeKB += A_LoopFileSizeKB
MsgBox Size of %WhichFolder% is %FolderSizeKB% KB.
```

```
; Example #3: Retrieve file names sorted by name
(see next example to sort by date):
```

```
FileList := "" ; Initialize to be blank.
```

```
Loop, Files, C:*.*
```

```
FileList .= A_LoopFileName "`n"
```

```
Sort, FileList, %FileList%, R ; The R option
sorts in reverse order. See Sort for other
options.
```

```
Loop, Parse, %FileList%, `n
```

```
{
```

```
if A_LoopField = "" ; Ignore the blank item
at the end of the list.
```

```
continue
```

```
Result := MsgBox("File number %A_Index% is
%A_LoopField%. Continue?",, 4)
```

```
if Result = "No"
```

```
break
```

```
}
```

```
; Example #4: Retrieve file names sorted by
modification date:
```

```
FileList := ""
```

```
Loop, Files, %A_MyDocuments%\Photos*.*, FD ;
```

```
Include Files and Directories
```

```
FileList .= A_LoopFileTimeModified "`t"
```

```
A_LoopFileName "`n"
```

```
Sort, FileList, %FileList% ; Sort by date.
```

```

Loop, Parse, %FileList%, `n
{
 if A_LoopField = "" ; Omit the last linefeed
 (blank item) at the end of the list.
 continue
 FileItem := StrSplit(A_LoopField, A_Tab) ;
 Split into two parts at the tab char.
 Result := MsgBox("The next file (modified at "
FileItem[1] ") is:`n" FileItem[2]
`n`nContinue?",, 4)
 if Result = "No"
 break
}

```

```

; Example #5: Copy only the source files that are
newer than their counterparts
; in the destination:
CopyIfNewer:
; Caller has set the variables CopySourcePattern
and CopyDest for us.
Loop, Files, %CopySourcePattern%
{
 copy_it := false
 if !FileExist(CopyDest "\" A_LoopFileName) ;
 Always copy if target file doesn't yet exist.
 copy_it := true
 else
 {
 FileGetTime, time,
%CopyDest%\%A_LoopFileName%
 time := DateDiff(A_Now,
A_LoopFileTimeModified, "Seconds") ; Subtract the
source file's time from the destination's.
 if time < 0 ; Source file is newer than

```

destination file.

```
 copy_it := true
 }
 if copy_it
 {
 FileCopy, %A_LoopFilePath%,
 %CopyDest%\%A_LoopFileName%, 1 ; Copy with
 overwrite=yes
 if ErrorLevel
 MsgBox, Could not copy
 "%A_LoopFilePath%" to
 "%CopyDest%\%A_LoopFileName%".
 }
}
Return
```

; Example #6: Convert filenames passed in via  
command-line parameters to long names,  
; complete path, and correct uppercase/lowercase  
characters as stored in the file system.

Loop, A\_Args.Length() ; For each parameter (or  
file dropped onto a script):

```
{
 GivenPath := A_Args[A_Index]
 Loop Files, %GivenPath%, FD ; Include files
 and directories.
 LongPath := A_LoopFilePath
 MsgBox The case-corrected long path name of
 file`n%GivenPath%`nis:`n%LongPath%
}
```

# Loop (read file contents)

Retrieves the lines in a text file, one at a time.

```
Loop Read, InputFile [, OutputFile]
```

## Parameters

### Read

This parameter must be the word READ, and cannot be an expression or variable reference.

### InputFile

The name of the text file whose contents will be read by the loop, which is assumed to be in %A\_WorkingDir% if an absolute path isn't specified. Windows and Unix formats are supported; that is, the file's lines may end in either carriage return and linefeed (`\r\n`) or just linefeed (`\n`).

### OutputFile

(Optional) The name of the file to be kept open for the duration of the loop, which is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

Within the loop's body, use the `FileAppend` command with only one parameter (the text to be written) to append to this special file. Appending to a file in this manner performs better than using `FileAppend` in its 2-

parameter mode because the file does not need to be closed and re-opened for each operation. Remember to include a linefeed (`\n`) after the text, if desired.

The file is not opened if nothing is ever written to it. This happens if the Loop performs zero iterations or if it never uses the `FileAppend` command.

**End of line (EOL) translation:** To disable EOL translation, prepend an asterisk to the filename. This causes each linefeed character (`\n`) to be written as a single linefeed (LF) rather than the Windows standard of CR+LF. For example: `*C:\My Unix File.txt`. Even without the asterisk, EOL translation is disabled automatically if the Loop's first use of `FileAppend` writes any carriage return and linefeed pairs (`\r\n`).

**Standard Output (stdout):** Specifying an asterisk (\*) for `OutputFile` sends any text written by `FileAppend` to standard output (stdout).

Although such output can be redirected to a file, piped to another EXE, or captured by fancy text editors, it will not appear at the command prompt it was launched from. See `FileAppend` for more details.

**Escaped Commas:** Unlike the last parameter of most other commands, commas in `OutputFile` must be escaped (`\,`).

## Remarks

A file-reading loop is useful when you want to operate on each line contained in

a text file, one at a time. The file is kept open for the entire operation to avoid having to re-scan each time to find the next line.

The built-in variable **A\_LoopReadLine** exists within any file-reading loop. It contains the contents of the current line excluding the carriage return and linefeed (`\r\n`) that marks the end of the line. If an inner file-reading loop is enclosed by an outer file-reading loop, the innermost loop's file-line will take precedence.

Lines up to 65,534 characters long can be read. If the length of a line exceeds this, its remaining characters will be read during the next loop iteration.

`StrSplit` or a [parsing loop](#) is often used inside a file-reading loop to parse the contents of each line retrieved from *InputFile*. For example, if *InputFile*'s lines are each a series of tab-delimited fields, those fields can individually retrieved as in this example:

```
Loop, read, C:\Database Export.txt
{
 Loop, parse, %A_LoopReadLine%, %A_Tab%
 {
 MsgBox, Field number %A_Index% is
%A_LoopField%.
 }
}
```

To load an entire file into variable, use [FileRead](#) because it performs much better than a loop (especially for large files).

To have multiple files open simultaneously, use `DllCall()` as shown in [this example](#).

See [Loop](#) for information about [Blocks](#), [Break](#), [Continue](#), and the `A_Index` variable (which exists in every type of loop).

To control how the file is decoded when no byte order mark is present, use [FileEncoding](#).

## Related

[FileEncoding](#), [FileOpen/File Object](#), [FileRead](#), [FileAppend](#), [Sort](#), [Loop](#), [Break](#), [Continue](#), [Blocks](#), [FileSetAttrib](#), [FileSetTime](#)

## Examples

```
; Example #1: Only those lines of the 1st file
that contain the word FAMILY will be written to
the 2nd file.
; Uncomment the first line to overwrite rather
than append to any existing file.
;FileDelete, C:\Docs\Family Addresses.txt
```

```
Loop, read, C:\Docs\Address List.txt,
C:\Docs\Family Addresses.txt
{
 if InStr(A_LoopReadLine, "family"),
FileAppend("%A_LoopReadLine%n")
}
```

```
; Example #2: Retrieve the last line from a text file.
```

```
Loop, read, C:\Log File.txt
```

```
 last_line := A_LoopReadLine ; When loop finishes, this will hold the last line.
```

```
; Example #3: A working script that attempts to extract all FTP and HTTP
```

```
; URLs from a text or HTML file:
```

```
FileSelect, SourceFile, 3,, Pick a text or HTML file to analyze.
```

```
if SourceFile = ""
```

```
 return ; This will exit in this case.
```

```
SplitPath, %SourceFile%,, SourceFilePath,, SourceFileNoExt
```

```
DestFile := "%SourceFilePath%\%SourceFileNoExt%
Extracted Links.txt"
```

```
if FileExist(DestFile)
```

```
{
```

```
 Result := MsgBox("Overwrite the existing links file? Press No to append to it.`n`nFILE:
```

```
%DestFile%",, 4)
```

```
 if Result = "Yes"
```

```
 FileDelete, %DestFile%
```

```
}
```

```
LinkCount := 0
```

```
Loop, read, %SourceFile%, %DestFile%
```

```
{
```

```
 URLSearchString := A_LoopReadLine
```

```
 Gosub, URLSearch
```

```
}
MsgBox %LinkCount% links were found and written to
"%DestFile%".
return
```

```
URLSearch:
```

```
; It's done this particular way because some URLs
have other URLs embedded inside them:
```

```
URLStart1 := InStr(URLSearchString, "http://")
```

```
URLStart2 := InStr(URLSearchString, "ftp://")
```

```
URLStart3 := InStr(URLSearchString, "www.")
```

```
; Find the left-most starting position:
```

```
URLStart := URLStart1 ; Set starting default.
```

```
Loop
```

```
{
```

```
; It helps performance (at least in a script
with many variables) to resolve
```

```
; "URLStart%A_Index%" only once:
```

```
ArrayElement := URLStart%A_Index%
```

```
if ArrayElement = "" ; End of the pseudo-
array has been reached.
```

```
break
```

```
if !ArrayElement ; This element is
disqualified.
```

```
continue
```

```
if !URLStart
```

```
URLStart := ArrayElement
```

```
else ; URLStart has a valid position in it, so
compare it with ArrayElement.
```

```
{
```

```
if ArrayElement
```

```
if ArrayElement < URLStart
```

```
URLStart := ArrayElement
```

```
}
```

```
}
```

```

if !URLStart ; No URLs exist in URLSearchString.
 return

; Otherwise, extract this URL:
URL := SubStr(URLSearchString, URLStart) ; Omit
the beginning/irrelevant part.
Loop, parse, %URL%, %A_Tab%%A_Space%<> ; Find the
first space, tab, or angle (if any).
{
 URL := A_LoopField
 break ; i.e. perform only one loop iteration
to fetch the first "field".
}
; If the above loop had zero iterations because
there were no ending characters found,
; leave the contents of the URL var untouched.

; If the URL ends in a double quote, remove it.
For now, StrReplace is used, but
; note that it seems that double quotes can
legitimately exist inside URLs, so this
; might damage them:
StrReplace, URLLCleansed, %URL%, `"`
FileAppend, %URLCleansed%`n
LinkCount += 1

; See if there are any other URLs in this line:
StrLen, CharactersToOmit, %URL%
CharactersToOmit += URLStart
URLSearchString := SubStr(URLSearchString,
CharactersToOmit)
Gosub, URLSearch ; Recursive call to self.
return

```

# SetWorkingDir

Changes the script's current working directory.

**SetWorkingDir** DirName

```
Command Example: SetWorkingDir A_ScriptDir
Function Example: SetWorkingDir(A_scriptDir)
```

## Parameters

### DirName

The name of the new working directory, which is assumed to be a subfolder of the current %A\_WorkingDir% if an absolute path isn't specified.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

The script's working directory is the default directory that is used to access files and folders when an absolute path has not been specified. In the following example, the file *My Filename.txt* is assumed to be in %A\_WorkingDir%:

```
FileAppend, A Line of Text, My Filename.txt.
```

The script's working directory defaults to `A_ScriptDir`, regardless of how the script was launched. By contrast, the value of `A_InitialWorkingDir` is determined by how the script was launched. For example, if it was run via shortcut -- such as on the Start Menu -- its initial working directory is determined by the "Start in" field within the shortcut's properties.

Once changed, the new working directory is instantly and globally in effect throughout the script. All interrupted, [paused](#), and newly launched [threads](#) are affected, including [Timers](#).

## Related

[A\\_WorkingDir](#), [A\\_InitialWorkingDir](#), [A\\_ScriptDir](#), [DirSelect](#)

## Example

```
SetWorkingDir %A_ScriptDir%
SetWorkingDir, D:\My Folder\Temp
```

# SplitPath

Separates a file name or URL into its name, directory, extension, and drive.

```
SplitPath Path [, OutFileName, OutDir, OutExtension,
OutNameNoExt, OutDrive]
```

## Parameters

### Path

The path or file name to be analyzed.

### OutFileName

Name of the variable in which to store the file name without its path. The file's extension is included.

### OutDir

Name of the variable in which to store the directory of the file, including drive letter or share name (if present). The final backslash is not included even if the file is located in a drive's root directory.

### OutExtension

Name of the variable in which to store the file's extension (e.g. TXT, DOC, or EXE). The dot is not included.

### OutNameNoExt

Name of the variable in which to store the file name without its path, dot and extension.

### OutDrive

Name of the variable in which to store the drive letter or server name of the file. If the file is on a local or mapped drive, the variable will be set to the drive letter followed by a colon (no backslash). If the file is on a network path (UNC), the variable will be set to the share name, e.g. \\Workstation01

### Remarks

Any of the output variables may be omitted if the corresponding information is not needed.

If *Path* contains a filename that lacks a drive letter (that is, it has no path or merely a relative path), *OutDrive* will be made blank but all the other output variables will be set correctly. Similarly, if there is no path present, *OutDir* will be made blank; and if there is a path but no file name present, *OutFileName* and *OutNameNoExt* will be made blank.

Actual files and directories in the file system are not checked by this command. It simply analyzes the provided string.

Wildcards (\* and ?) and other characters illegal in filenames are treated the same as legal characters, with the exception of colon, backslash, and period (dot), which are processed according to their nature in delimiting the drive letter,

directory, and extension of the file.

**Support for URLs:** If *Path* contains a colon-double-slash, such as `http://domain.com` or `ftp://domain.com`, *OutDir* is set to the protocol prefix + domain name + directory (e.g. `http://domain.com/images`) and *OutDrive* is set to the protocol prefix + domain name (e.g. `http://domain.com`). All other variables are set according to their definitions above.

## Related

[A\\_LoopFileExt](#), [StrSplit](#), [InStr](#), [SubStr](#), [FileSelect](#), [DirSelect](#)

## Example

```
FullFileName := "C:\My Documents\Address List.txt"

; To fetch only the bare filename from the above:
SplitPath, %FullFileName%, name

; To fetch only its directory:
SplitPath, %FullFileName%, , dir

; To fetch all info:
SplitPath, %FullFileName%, name, dir, ext,
name_no_ext, drive

; The above will set the variables as follows:
; name = Address List.txt
; dir = C:\My Documents
; ext = txt
; name_no_ext = Address List
; drive = C:
```

# #Include / #IncludeAgain

Causes the script to behave as though the specified file's contents are present at this exact position.

```
#Include FileOrDirName
#Include <LibName>
#IncludeAgain FileOrDirName
```

## Parameters

### FileOrDirName

The path of a file or directory as explained below. This **must not** contain double quotes, wildcards, or variable references except %A\_ScriptDir%, %A\_AppData%, %A\_AppDataCommon% and %A\_LineFile%. Escape sequences other than semicolon (;) must not be used, nor are they needed because characters such as percent signs are treated literally.

**File:** The name of the file to be included, which is assumed to be in the same directory as the current file if an absolute path is not specified.

Note: [SetWorkingDir](#) has no effect on #Include because #Include is processed before the script begins executing.

**Directory:** Specify a directory instead of a file to change the working directory used by all subsequent occurrences of #Include and [FileInstall](#) in the current file. Note: Changing the working directory in this way does

not affect the script's initial working directory when it starts running (`A_WorkingDir`). To change that, use `SetWorkingDir` at the top of the script.

### LibName

A library file or function name. For example, `#include <lib>` and `#include <lib_func>` would both include `lib.ahk` from one of the function library folders.

### Remarks

A script behaves as though the included file's contents are physically present at the exact position of the `#Include` directive (as though a copy-and-paste were done from the included file). Consequently, it generally cannot merge two isolated scripts together into one functioning script (to achieve that, see [www.autohotkey.com/forum/topic18545.html](http://www.autohotkey.com/forum/topic18545.html)).

`#Include` ensures that *FileName* is included only once, even if multiple inclusions are encountered for it. By contrast, `#IncludeAgain` allows multiple inclusions of the same file, while being the same as `#Include` in all other respects.

The *FileName* parameter may optionally be preceded by `*i` and a single space, which causes the program to ignore any failure to read the included file. For example: `#Include *i SpecialOptions.ahk`. This option should be used only when the included file's contents are not essential to the main script's operation.

Lines displayed in the main window via [ListLines](#) or the menu View->Lines are always numbered according to their physical order within their own files. In other words, including a new file will change the line numbering of the main script file by only one line, namely that of the `#Include` line itself (except for [compiled scripts](#), which merge their included files into one big script at the time of compilation).

`#Include` is often used to load [functions](#) defined in an external file. Unlike subroutine labels, [functions](#) can be included at the very top of the script without affecting the [auto-execute section](#).

Like other `#` directives, `#Include` cannot be executed conditionally. In other words, this example would not work:

```
if x = 1
 #Include SomeFile.ahk ; This line takes
effect regardless of the value of x.
 y := 2 ; And this line would belong to the
IF above, since # directives cannot belong to
IFs.
```

Files can be automatically included (i.e. without having to use `#Include`) by calling a [library function](#) by name.

## Related

[Libraries of Functions](#), [Functions](#), [FileInstall](#)

## Example

```
#Include C:\My Documents\Scripts\Utility
Subroutines.ahk
#Include %A_ScriptDir% ; Changes the working
directory for subsequent #Includes and
FileInstalls.
#Include C:\My Scripts ; Same as above but for an
explicitly named directory.
```

## {...} (block)

A pair of braces denotes a block. Blocks are typically used with [functions](#), [Else](#), [Loop](#), [While-loop](#), and [IF-commands](#).

```
{
zero or more commands
}
```

### Remarks

A block is used to bind two or more commands together. It can also be used to change which IF an [ELSE](#) belongs to, as in this example where the block forces the [ELSE](#) to belong to the first IF rather than the second:

```
if var1 = 1
{
 if var2 = "abc"
 sleep, 1
}
else
 return
```

Although blocks can be used anywhere, currently they are only meaningful when used with [functions](#), [Else](#), [Loop](#), or [if-statements](#).

If an [IF](#), [ELSE](#), [Loop](#), [While-loop](#), or [For-loop](#) has only a single command, that command need not be enclosed in a block. However, there may be cases where

doing so enhances the readability or maintainability of the script.

A block may be empty (contain zero commands), which may be useful in cases where you want to comment out the contents of the block without removing the block itself.

**One True Brace (OTB, K&R style):** The OTB style may optionally be used in the following places: [if-statements](#), the [else](#) keyword, [while-loops](#), [For-loops](#), [normal loops](#), [function definitions](#), [Try](#), and [Catch](#). This style puts the block's opening brace on the same line as the block's controlling statement rather than underneath on a line by itself. For example:

```
if x < y {
 ...
} else {
 ...
}
while x < y {
 ...
}
for k,v in obj {
 ...
}
loop RepeatCount {
 ...
}
myFunction(x, y) {
 ...
}
try {
 ...
} catch e {
 ...
}
```

```
} Finally {

}
```

Similarly, a command or other action may exist to the right of a brace (except the open-brace of the One True Brace style). For example:

```
if x = 1
{ MsgBox This line appears to the right of an
 opening brace. It executes whenever the IF-
 statement is true.
 MsgBox This is the next line.
} MsgBox This line appears to the right of a
closing brace. It executes unconditionally.
```

## Related

[Functions](#), [While-loop](#), [Loop](#), [Else, If \(Expression\)](#)

## Example

```
if x = 1
{
 MsgBox, test1
 Sleep, 5
}
else
 MsgBox, test2
```

# Break

Exits (terminates) a [loop](#). Valid inside any kind of [loop](#).

```
Break [LoopLabel]
```

If specified, *LoopLabel* identifies which loop this statement should apply to; either by [label name](#) or numeric nesting level. If omitted or 1, this statement applies to the innermost loop in which it is enclosed. *LoopLabel* must be a constant value - variables and expressions are not supported. If a [label](#) is specified, it must point directly at a loop command.

The use of [Break](#) and [Continue](#) are encouraged over [goto](#) since they usually make scripts more readable and maintainable.

## Related

[Continue](#), [Loop](#), [While-loop](#), [For-loop](#), [Blocks](#), [Labels](#)

## Example

```
Loop
{
 ...
 if var > 25
 break
 ...
 if var <= 5
 continue
```

```
}
```

```
; Break the outer loop from within a nested loop.
```

```
outer:
```

```
Loop 3
```

```
{
```

```
 x := A_Index
```

```
 Loop 3
```

```
 {
```

```
 if (x*A_Index = 6)
```

```
 break outer ; Equivalent to break 2
```

```
or goto break_outer.
```

```
 MsgBox("%x%,%A_Index%")
```

```
 }
```

```
}
```

```
break_outer: ; For goto.
```

# Catch

Specifies the code to execute if an exception is raised during execution of a `try` statement.

```
Catch [OutputVar]
 Statement
```

```
Catch [OutputVar]
{
 Statements
}
```

## Parameters

### OutputVar

(Optional) The name of the variable in which to store the value of the exception.

### Statement(s)

The commands or expressions to execute if an exception is raised.

## Remarks

Every use of `catch` must belong to (be associated with) a `try` statement above it. A `catch` always belongs to the nearest unclaimed `try` statement above it unless a `block` is used to change that behavior.

The [One True Brace \(OTB\)](#) style may optionally be used. For example:

```
try {
 ...
} catch e {
 ...
}
```

## Runtime Errors

A *try-catch* statement can also be used to handle runtime errors. If a runtime error or exception is not handled, an error message is displayed and the current thread exits. Loadtime errors cannot be handled, since they occur before the *try* statement is executed.

The value that is stored in *OutputVar* (if present) is an exception [object](#) that contains the following fields:

**What:** The name of the command or function which was executing or about to execute when the error occurred.

**File:** The full path of the script file which contains the line at which the error occurred.

**Line:** The line number at which the error occurred.

**Message:** An error message.

**Extra:** Additional information about the error, if available.

**Note:** These details should be used only for debugging purposes as they may change in a future version.

## Related

[Try](#), [Throw](#), [Finally](#), [Blocks](#)

## Examples

See [Try](#).

# Continue

Skips the rest of the current [loop](#) iteration and begins a new one. Valid inside any kind of [loop](#).

```
Continue [LoopLabel]
```

If specified, [LoopLabel](#) identifies which loop this statement should apply to; either by [label name](#) or numeric nesting level. If omitted or 1, this statement applies to the innermost loop in which it is enclosed. [LoopLabel](#) must be a constant value - variables and expressions are not supported. If a [label](#) is specified, it must point directly at a loop command.

Continue behaves the same as reaching the loop's closing brace:

1. It increases [A\\_Index](#) by 1.
2. It skips the rest of the loop's body.
3. The loop's condition (if it has one) is checked to see if it is satisfied. If so, a new iteration begins; otherwise the loop ends.

The use of [Break](#) and [Continue](#) are encouraged over [goto](#) since they usually make scripts more readable and maintainable.

## Related

[Break](#), [Loop](#), [Until](#), [While-loop](#), [For-loop](#), [Blocks](#), [Labels](#)

## Example

```
; This example displays 5 MsgBoxes, one for each
number between 6 and 10.
; Note that in the first 5 iterations of the Loop,
the "continue" command
; causes the loop to start over before it reaches
the MsgBox line.
```

```
Loop, 10
{
 if A_Index <= 5
 continue
 MsgBox %A_Index%
}
```

```
; Continue the outer loop from within a nested
loop.
```

```
outer:
Loop 3
{
 x := A_Index
 Loop 3
 {
 if (x*A_Index = 4)
 continue outer ; Equivalent to
 continue 2 or goto continue_outer.
 MsgBox("%x%,%A_Index%")
 }
 continue_outer: ; For goto.
}
```

# Critical

Prevents the [current thread](#) from being interrupted by other threads, or enables it to be interrupted.

```
Critical ["Off"]
```

```
Critical 50 ; See bottom of remarks.
```

If the first parameter is omitted (or the word On), the [current thread](#) is made critical, meaning that it cannot be interrupted by another thread. If the first parameter is the word Off or the number 0, the current thread immediately becomes interruptible, regardless of the settings of [Thread Interrupt](#).

## Behavior of Critical Threads

Unlike [high-priority](#) threads, events that occur during a critical thread are not discarded. For example, if the user presses a [hotkey](#) while the current thread is critical, the hotkey is buffered indefinitely until the current thread finishes or becomes noncritical, at which time the hotkey is launched as a new thread.

A critical thread will be interrupted in emergencies. Emergencies consist of: 1) the [OnExit](#) callback function; 2) any [OnMessage](#) function that monitors a message number less than 0x312 (or a [callback](#) triggered by such a message); and 3) any [callback](#) indirectly triggered by the critical thread itself (e.g. via [SendMessage](#) or [DllCall](#)). To avoid these interruptions, temporarily disable such functions.

A critical thread becomes interruptible when a [MsgBox](#) or other dialog is displayed. However, unlike [Thread Interrupt](#), the thread becomes critical again after the user dismisses the dialog.

## Critical Off

When buffered events are waiting to start new threads, using `Critical Off` will not result in an immediate interruption of the current thread. Instead, an average of 5 milliseconds will pass before an interruption occurs. This makes it more than 99.999% likely that at least one line after `Critical Off` will execute before an interruption. You can force interruptions to occur immediately by means of a delay such as a `Sleep -1` or a [WinWait](#) for a window that does not yet exist.

`Critical Off` cancels the current thread's period of uninterruptibility even if the thread was not Critical, thereby letting events such as [Size](#) be processed sooner or more predictably.

## Thread Settings

See [A\\_IsCritical](#) for how to save and restore the current setting of Critical. However, since Critical is a thread-specific setting, when a critical thread ends, the underlying/resumed thread (if any) will be automatically noncritical. Consequently there is no need to do "Critical Off" right before ending a thread.

If Critical is not used in the auto-execute section (top part of the script), all threads start off as noncritical (though the settings of [Thread Interrupt](#) will still

apply). By contrast, if the auto-execute section turns on Critical but never turns it off, every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off critical.

The command [Thread NoTimers](#) is similar to Critical except that it only prevents interruptions from [timers](#).

## Message Check Interval

Specifying a positive number as the first parameter (e.g. `Critical 30`) turns on Critical but also changes the number of milliseconds between checks of the internal message queue. If unspecified, messages are checked every 16 milliseconds while Critical is On, and every 5 ms while Critical is Off.

Increasing the interval postpones the arrival of messages/events, which gives the [current thread](#) more time to finish. This reduces the possibility that certain [OnMessage\(\)](#) and [GUI events](#) will be lost due to "thread already running".

However, commands that wait such as [Sleep](#) and [WinWait](#) will check messages regardless of this setting (a workaround is `DllCall("Sleep", UInt, 500)`). Note: Increasing the message-check interval too much may reduce the responsiveness of various events such as [GUI window repainting](#).

In v1.0.47+, specifying a positive number as the first parameter (e.g. `Critical 30`) turns on Critical but also changes the number of milliseconds between checks of the internal message queue. If unspecified, messages are checked every 16 milliseconds while Critical is On, and every 5 ms while Critical is Off. Increasing the interval postpones the arrival of messages/events, which gives the [current thread](#) more time to finish. This reduces the possibility

that certain `OnMessage()` and GUI events will be lost due to "thread already running". However, commands that wait such as `Sleep` and `WinWait` will check messages regardless of this setting (a workaround is `DllCall("Sleep", UInt, 500)`). Note: Increasing the message-check interval too much may reduce the responsiveness of various events such as GUI window repainting.

## Related

[Thread \(command\)](#), [Threads](#), [#MaxThreadsPerHotkey](#), [#MaxThreadsBuffer](#), [OnMessage](#), [RegisterCallback](#), [Hotkey](#), [Menu](#), [SetTimer](#)

## Example

```
#space:: ; Win+Space hotkey.
Critical
ToolTip No new threads will launch until after
this ToolTip disappears.
Sleep 3000
ToolTip ; Turn off the tip.
return ; Returning from a hotkey subroutine ends
the thread. Any underlying thread to be resumed is
noncritical by definition.
```

# Else

Specifies the command(s) to perform if an IF-statement evaluates to FALSE. When more than one command is present, enclose them in a [block](#) (braces).

## Else

## Remarks

Every use of ELSE must belong to (be associated with) an IF-statement above it. An ELSE always belongs to the nearest unclaimed IF-statement above it unless a [block](#) is used to change that behavior.

An ELSE can be followed immediately by any other single command on the same line. This is most often used for "else if" ladders (see examples at the bottom).

When an IF or an ELSE owns more than one line, those lines must be enclosed in braces. However, if only one line belongs to an IF or ELSE, the braces are optional. For example:

```
if count > 0 ; No braces are required around
the next line because it's only a single line.
 MsgBox Press OK to begin the process.
else ; Braces must be used around the section
below because it consists of more than one
line.
{
 WinClose Untitled - Notepad
```

```
 MsgBox There are no items present.
}
```

The [One True Brace \(OTB\) style](#) may optionally be used around an "else". For example:

```
if IsDone {
 ...
} else if (x < y) {
 ...
} else {
 ...
}
```

## Related

[Blocks, if \(expression\)](#)

## Examples

```
If WinExist("Untitled - Notepad")
{
 WinActivate
 Send This is a test.{Enter}
}
else
{
 WinActivate, Some Other Window
 MouseClick, left, 100, 200
}

if x = 1
```

```
Gosub, a1
else if x = 2 ; "else if" style
 Gosub, a2
else if x = 3
{
 Gosub, a3
 Sleep, 1
}
else Gosub, a4 ; i.e. Any single command can be
on the same line with an ELSE.

; Also OK:
if y = 1, Gosub, b1
else {
 Sleep, 1
 Gosub, b2
}
```

# Exit

Exits the [current thread](#).

```
Exit [ExitCode]
```

```
Command Example: Exit 100
```

## Parameters

### ExitCode

An integer between -2147483648 and 2147483647 that is returned to its caller when the script exits. This code is accessible to any program that spawned the script, such as another script (via `RunWait`) or a batch (.bat) file. If omitted, *ExitCode* defaults to zero. Zero is traditionally used to indicate success.

## Remarks

If the script is not [persistent](#) and this is the last thread, the entire script exits.

Otherwise, the `Exit` command terminates the [current thread](#). In other words, the stack of subroutines called directly or indirectly by a [menu](#), [timer](#), or [hotkey](#) subroutine will all be returned from as though a [Return](#) were immediately encountered in each. If used directly inside such a subroutine -- rather than in one of the subroutines called indirectly by it -- `Exit` is equivalent to [Return](#).

Use `ExitApp` to completely terminate a script that is `persistent`.

## Related

`ExitApp`, `OnExit`, `Functions`, `Gosub`, `Return`, `Threads`, `#Persistent`

## Example

```
#Z::
Gosub, Sub2
MsgBox, This msgbox will never happen because of
the EXIT.
return

Sub2:
Exit ; Terminate this subroutine as well as the
calling subroutine.
```

# ExitApp

Terminates the script unconditionally.

```
ExitApp [ExitCode]
```

## Parameters

### ExitCode

An integer between -2147483648 and 2147483647 that is returned to its caller when the script exits. This code is accessible to any program that spawned the script, such as another script (via RunWait) or a batch (.bat) file. If omitted, *ExitCode* defaults to zero. Zero is traditionally used to indicate success.

## Remarks

This is equivalent to choosing "Exit" from the script's tray menu or main menu.

Any [OnExit](#) functions registered by the script are called before the script terminates. If an OnExit function returns a non-zero integer, the script does not terminate; instead, the current [thread](#) exits as if [Exit](#) was called.

ExitApp is often unnecessary in scripts which are not [persistent](#).

## Related

Exit, OnExit, #Persistent

## Example

```
#x::ExitApp ; Assign a hotkey to terminate this
script.
```

# Finally

Ensures that one or more statements (commands or expressions) are always executed after the execution of a `try` statement.

```
Finally Statement
```

```
Finally
{
 Statements
}
```

## Remarks

Every use of *finally* must belong to (be associated with) a `try` (or `catch`) statement above it. A *finally* always belongs to the nearest unclaimed `try` statement above it unless a `block` is used to change that behavior.

If a *finally* statement is applied to a `try` statement with no `catch` block, the exception is not absorbed by the latter and exception propagation continues after the execution of the *finally* statement.

It is not allowed to use *goto/break/continue* in order to exit a *finally* statement, as that would imply breaking exception propagation (however, normal usage that keeps execution inside the block is allowed). This mistake is detected at load time (and at run time for dynamic *goto* statements).

The [One True Brace \(OTB\)](#) style may optionally be used with the *finally*

command. For example:

```
try {
 ...
} finally {
 ...
}

try {
 ...
} catch e {
 ...
} finally {
 ...
}
```

## Related

[Try, Catch, Throw, Blocks](#)

## Examples

```
try
{
 Tooltip, Working...
 Example1()
}
catch e
{
 ; For more detail about the object that e
contains, see Catch.
 MsgBox("Exception thrown!\n\nwhat: " e.what
"\nfile: " e.file
```

```
 . "\nline: " & e.line & "\nmessage: " &
e.message & "\nextra: " & e.extra, 16)
 }
 finally
 {
 ToolTip ; hide the ToolTip
 }
}
```

MsgBox, Done!

**; This function has a Finally block that acts as cleanup code**

```
Example1()
{
 try
 Example2()
 finally
 MsgBox, This is always executed regardless
of exceptions
}
}
```

**; This function fails when the minutes are odd**

```
Example2()
{
 if Mod(A_Min, 2)
 throw Exception("Test exception")
 MsgBox, Example2 did not fail
}
}
```

# For-loop

Repeats a series of commands once for each key-value pair in an object.

```
For Key [, Value] in Expression
```

## Parameters

### Key

### Value

The names of the variables in which to store the key and value at the beginning of each iteration.

When the loop breaks or completes, these variables are restored to their former values.

### Expression

An [expression](#) which results in an object, or a variable which contains an object.

## Remarks

*Expression* is evaluated only once, before the loop begins. If its result is not an object, execution jumps immediately to the line following the loop's body; otherwise, the object's `__NewEnum()` method is called to retrieve an *enumerator* object. At the beginning of each iteration, the enumerator's [Next](#)

method is used to retrieve the next key-value pair. If `Next()` returns zero or an empty string, the loop terminates.

Although not exactly equivalent to a for-loop, the following demonstrates this process:

```
_enum := (Expression)._NewEnum()
if IsObject(_enum)
 while _enum.Next(Key, Value)
 {
 ...
 }
```

Existing key-value pairs may be modified during the loop, but inserting or removing keys may cause some items to be skipped or enumerated multiple times. One workaround is to build a list of keys to remove, then use a second loop to remove the keys after the first loop completes. Note that `Object.Delete(first, last)` can be used to remove a range of keys without looping.

A for-loop is usually followed by a `block`, which is a collection of statements that form the *body* of the loop. However, a loop with only a single statement does not require a block (an "if" and its "else" count as a single statement for this purpose). The One True Brace (OTB) style may optionally be used, which allows the open-brace to appear on the same line rather than underneath. For example: `for x, y in z {`.

As with all loops, `Break`, `Continue` and `A_Index` may be used.

## COM Objects

Since *Key* and *Value* are passed directly to the enumerator's `Next()` method, the values they are assigned depends on what type of object is being enumerated. For COM objects, *Key* contains the value returned by `IEnumVARIANT::Next` and *Value* contains a number which represents its [variant type](#). For example, when used with a `Scripting.Dictionary` object, each *Key* contains a key from the dictionary and *Value* is typically 8 for strings and 3 for integers. See [ComObjType](#) for a list of type codes.

When enumerating a `SafeArray`, *Key* contains the current element and *Value* contains its variant type.

## Related

[Enumerator object](#), [Object.\\_NewEnum](#), [While-loop](#), [Loop](#), [Until](#), [Break](#), [Continue](#), [Blocks](#)

## Examples

```
; List the key-value pairs of an object:
colours := Object("red", 0xFF0000, "blue",
0x0000FF, "green", 0x00FF00)
; The above expression could be used directly in
place of "colours" below:
for k, v in colours
 s := k "=" v "`n"
MsgBox % s
```

```
; List all open Explorer and Internet Explorer
windows:
for window in
ComObjCreate("Shell.Application").Windows
 windows .= window.LocationName " :: "
window.LocationURL "`n"
MsgBox % windows
```

```
/*
```

```
Class: CEnumerator
```

Generic enumerator object that can be used for iterating over numeric keys.

The array must not be modified during iteration, otherwise the iterated range will be invalid.

It's possible to define a custom Length() function for array boundaries.

If there are missing array members between 1 and max index, they will be iterated but will have a value of "".

This means that real sparse arrays are not supported by this enumerator by design.

To make an object use this iterator, insert this function in the class definition:

```
 _NewEnum()
 {
 return new CEnumerator(this)
 }
```

Source:

<http://www.autohotkey.com/board/topic/2667-suggestions-on-documentation-improvements/?p=531509>

```
*/
```

```
; Iterate over the enumerator
```

```
For k, v in Test
 MsgBox %k%=%v%
```

```
; Test class for demonstrating usage
```

```
class Test
{
 static Data := ["abc", "def", "ghi"]

 _NewEnum()
 {
 return new CEnumerator(this.Data)
 }
}
```

```
class CEnumerator
{
```

```
 __New(Object)
 {
 this.Object := Object
 this.first := true
```

```
; Cache for speed. Useful if
custom Length() functions have poor performance.
; In return, that means that no
key-value pairs may be inserted during iteration
or the range will become invalid.
```

```
 this.ObjMaxIndex :=
Object.Length()
 }
```

```
 Next(ByRef key, ByRef value)
 {
 if (this.first)
 {
 this.Delete("first")
 key := 1
 }
 }
```



# Gosub

Jumps to the specified label and continues execution until [Return](#) is encountered.

**Gosub** Label

|                 |                 |                 |
|-----------------|-----------------|-----------------|
| <b>Command</b>  | <b>Example:</b> | Gosub "MyLabel" |
| <b>Function</b> | <b>Example:</b> | Gosub("Label")  |

## Parameters

### Label

The name of the [label](#), [hotkey label](#), or [hotstring label](#) to which to jump, which causes the commands beneath *Label* to be executed until a [Return](#) or [Exit](#) is encountered. "Return" causes the script to jump back to the first command beneath the Gosub and resume execution there. "Exit" terminates the [current thread](#).

## Remarks

*Label* can be a variable or expression only if parentheses are used. For example, `Gosub MySub` and `Gosub("MySub")` both execute the `MySub:` subroutine.

Performance is slightly reduced when using a dynamic label (that is, a variable or expression which returns a label name) because the target label must be

"looked up" each time rather than only once when the script is first loaded. An error dialog will be displayed if the label does not exist. To avoid this, call `IsLabel()` beforehand. For example:

```
if IsLabel(VarContainingLabelName)
 Gosub(VarContainingLabelName)
```

Although `Gosub` is useful for simple, general purpose subroutines, consider using [functions](#) for more complex purposes.

## Related

[Return](#), [Functions](#), [IsLabel](#), [Blocks](#), [Loop](#), [Goto](#)

## Example

```
Gosub, Label1
MsgBox "The Label1 subroutine has returned (it is
finished)."
```

```
return

Label1:
MsgBox "The Label1 subroutine is now running."
return
```

# Goto

Jumps to the specified label and continues execution.

**Goto** Label

|                          |                              |
|--------------------------|------------------------------|
| <b>Command Example:</b>  | <code>Goto "MyLabel"</code>  |
| <b>Function Example:</b> | <code>Goto("MyLabel")</code> |

## Parameters

### Label

The name of the [label](#) to which to jump.

## Remarks

*Label* can be a variable or expression only if parentheses are used. For example, `Goto MyLabel` and `Gosub("MyLabel")` both jump to `MyLabel:`.

Performance is slightly reduced when using a dynamic label (that is, a variable or expression which returns a label name) because the target label must be "looked up" each time rather than only once when the script is first loaded. An error dialog will be displayed if the label does not exist. To avoid this, call `IsLabel()` beforehand. For example:

```
if IsLabel(VarContainingLabelName)
 Goto(VarContainingLabelName)
```

The use of `Goto` is discouraged because it generally makes scripts less readable and harder to maintain. Consider using [Else](#), [Blocks](#), [Break](#), and [Continue](#) as substitutes for `Goto`.

## Related

[Gosub](#), [Return](#), [IsLabel](#), [Else](#), [Blocks](#), [Break](#), [Continue](#)

## Example

```
Goto, MyLabel
...
MyLabel:
Sleep, 100
...
```

# if (expression)

Specifies the command(s) to perform if an `expression` evaluates to TRUE.

```
if (expression)
```

## Remarks

If the if-statement's expression evaluates to true (which is any result other than an empty string or the number 0), the line or `block` underneath it is executed. Otherwise, if there is a corresponding ELSE, execution jumps to the line or block underneath it.

When an IF or an ELSE owns more than one line, those lines must be enclosed in `braces`. However, if only one line belongs to an IF or ELSE, the braces are optional. See the examples at the bottom of this page.

The space after `if` is optional if the expression starts with an open-parenthesis, as in `if(expression)`.

The One True Brace (OTB) style may optionally be used. For example:

```
if (x < y) {
 ...
}
if WinExist("Untitled - Notepad") {
 WinActivate
}
if IsDone {
```

```
 ...
} else {
 ...
}
```

Any type of statement can appear immediately to the right of an if-statement, but unlike an else-statement, a delimiting comma is required. For example:

```
if true, MsgBox ; Good
if true MsgBox ; Bad
```

## Related

[Expressions](#), [Assign expression \(:=\)](#), [Ternary operator \(a?b:c\)](#), [Blocks](#), [Else](#), [While-loop](#)

## Example

```
if (A_Index > 100)
 return

if (A_TickCount - StartTime > 2*MaxTime + 100)
{
 MsgBox Too much time has passed.
 ExitApp
}

if (Color = "Blue" or Color = "White")
{
 MsgBox The color is one of the allowed values.
 ExitApp
}
```



# Loop (normal)

Perform a series of commands repeatedly: either the specified number of times or until `break` is encountered.

**Loop** `[Count]`

## Parameters

### Count

How many times (iterations) to perform the loop, which can be an `expression`. If omitted, the Loop continues indefinitely until a `break` or `return` is encountered. However, an explicit blank value or number less than 1 causes the loop to be skipped entirely.

If *Count* is an expression, it is evaluated only once, right before the loop begins.

## Remarks

The loop command is usually followed by a `block`, which is a collection of statements that form the *body* of the loop. However, a loop with only a single statement does not require a block (an "if" and its "else" count as a single statement for this purpose).

A common use of this command is an infinite loop that uses the `break` command somewhere in the loop's *body* to determine when to stop the loop.

The use of `break` and `continue` inside a loop are encouraged as alternatives to `goto`, since they generally make a script more understandable and maintainable. To create a "Do...While" loop, make the last statement of the loop's *body* an IF statement that conditionally issues the `break` command. The same technique can be used to create a "While" loop, but the `While` command should be used instead.

The built-in variable `A_Index` contains the number of the current loop iteration. It contains 1 the first time the loop's *body* is executed. For the second time, it contains 2; and so on. If an inner loop is enclosed by an outer loop, the inner loop takes precedence. `A_Index` works inside all types of loops, including `file-loops` and `registry-loops`; but `A_Index` contains 0 outside of a loop.

`A_Index` can be assigned any integer value by the script. If *Count* is specified, changing `A_Index` affects the number of iterations that will be performed. For example, `A_Index := 3` would make the loop command act as though it is on the third iteration (`A_Index` will be 4 on the next iteration), while `A_Index-` would prevent the current iteration from being counted toward the total.

The `One True Brace (OTB)` style may optionally be used with normal loops (but specialized loops such as `file-pattern` and `parsing` support OTB only in expression mode). For example:

```
Loop {
 ...
}
Loop RepeatCount {
 ...
```

```
}
```

Specialized loops: Loops can be used to automatically retrieve files, folders, or registry items (one at a time). See [file-loop](#) and [registry-loop](#) for details. In addition, [file-reading loops](#) can operate on the entire contents of a file, one line at a time. Finally, [parsing loops](#) can operate on the individual fields contained inside a delimited string.

## Related

[Until](#), [While-loop](#), [For-loop](#), [Files-and-folders loop](#), [Registry loop](#), [File-reading loop](#), [Parsing loop](#), [Break](#), [Continue](#), [Blocks](#)

## Examples

```
Loop, 3
{
 MsgBox, Iteration number is %A_Index%. ;
 A_Index will be 1, 2, then 3
 Sleep, 100
}

Loop
{
 if a_index > 25
 break ; Terminate the loop
 if a_index < 20
 continue ; Skip the below and start a new
iteration
 MsgBox, a_index = %a_index% ; This will
display only the numbers 20 through 25
}
```



# Loop (parse a string)

Retrieves substrings (fields) from a string, one at a time.

```
Loop Parse, String [, Delimiters, OmitChars]
```

## Parameters

### Parse

This parameter must be the word PARSE, and cannot be an expression or variable reference.

### String

The string to analyze.

### Delimiters

If this parameter is blank or omitted, each character of the input string will be treated as a separate substring.

If this parameter is **CSV**, the string will be parsed in standard comma separated value format. Here is an example of a CSV line produced by MS Excel: "first field",SecondField,"the word ""special"" is quoted literally",,"last field, has literal comma"

Otherwise, *Delimiters* contains one or more characters (case sensitive), each of which is used to determine where the boundaries between substrings occur.

Delimiter characters are not considered to be part of the substrings themselves. In addition, if there is nothing between a pair of delimiters within the input string, the corresponding substring will be empty.

For example: `, (an escaped comma) would divide the string based on every occurrence of a comma. Similarly, %A\_Tab%%A\_Space% would start a new substring every time a space or tab is encountered in the input string.

To use a string as a delimiter rather than a character, first use [StrReplace](#) to replace all occurrences of the string with a single character that is never used literally in the text, e.g. one of these special characters: ¢  
¥|§©ª«®µ¶. Consider this example, which uses the string <br> as a delimiter:

```
StrReplace, NewHTML, %HTMLString%,
, ¢
Loop, parse, %NewHTML%, ¢ ; Parse the
string based on the cent symbol.
{
...
}
```

## OmitChars

An optional list of characters (case sensitive) to exclude from the beginning and end of each substring. For example, if *OmitChars* is %A\_Space%%A\_Tab%, spaces and tabs will be removed from the beginning and end (but not the middle) of every retrieved substring.

If *Delimiters* is blank, *OmitChars* indicates which characters should be excluded from consideration (the loop will not see them).

Unlike the last parameter of most other commands, commas in *OmitChars* must be escaped (`\,`).

## Remarks

A string parsing loop is useful when you want to operate on each field contained in a string, one at a time. Parsing loops use less memory than [StrSplit](#) (though either way the memory use is temporary) and in most cases they are easier to use.

The built-in variable **A\_LoopField** exists within any parsing loop. It contains the contents of the current substring (field). If an inner parsing loop is enclosed by an outer parsing loop, the innermost loop's field will take precedence.

Although there is no built-in variable "A\_LoopDelimiter", the example at the very bottom of this page demonstrates how to detect which delimiter was encountered for each field.

There is no restriction on the size of the input string or its fields.

To arrange the fields in a different order prior to parsing, use the [Sort](#) command.

See [Loop](#) for information about [Blocks](#), [Break](#), [Continue](#), and the `A_Index` variable (which exists in every type of loop).

## Related

StrSplit, file-reading loop, Loop, Break, Continue, Blocks, Sort, FileSetAttrib, FileSetTime

## Examples

**; Example #1:**

```
Colors := "red,green,blue"
Loop, parse, %Colors%, `
{
 MsgBox, Color number %A_Index% is
%A_LoopField%.
}
```

**; Example #2: Read the lines inside a variable, one by one (similar to a file-reading loop).**

**; A file can be loaded into a variable via**

**FileRead:**

```
Loop, parse, %FileContents%, `n, `r ; Specifying
`n prior to `r allows both Windows and Unix files
to be parsed.
{
 Result := MsgBox("Line number %A_Index% is
%A_LoopField%.`n`nContinue?",, 4)
 if Result = "No", break
}
```

**; Example #3: This is the same as the example above except that it's for the clipboard.**

```
; It's useful whenever the clipboard contains
files, such as those copied from an open
; Explorer window (the program automatically
converts such files to their file names):
```

```
Loop, parse, %clipboard%, `n, `r
{
 Result := MsgBox("File number %A_Index% is
%A_LoopField%.`n`nContinue?", , 4)
 if Result = "No", break
}
```

```
; Example #4: Parse a comma separated value (CSV)
file:
```

```
Loop, read, C:\Database Export.csv
{
 LineNumber := A_Index
 Loop, parse, %A_LoopReadLine%, CSV
 {
 Result := MsgBox("Field %LineNumber%-
%A_Index% is:`n%A_LoopField%`n`nContinue?", , 4)
 if Result = "No"
 return
 }
}
```

```
; Example #5: Determining which delimiter was
encountered.
```

```
; Initialize string to search.
```

```
Colors := "red,green|blue;yellow|cyan,magenta"
```

```
; Initialize counter to keep track of our position
in the string.
Position := 0

Loop, Parse, %Colors%, `,|;
{
 ; Calculate the position of the delimiter at
the end of this field.
 Position += StrLen(A_LoopField) + 1
 ; Retrieve the delimiter found by the parsing
loop.
 Delimiter := SubStr(Colors, Position, 1)

 MsgBox Field: %A_LoopField%`nDelimiter:
%Delimiter%
}
```

# Loop (registry)

Retrieves the contents of the specified registry subkey, one item at a time.

```
Loop Reg, KeyName [, Mode]
```

## Parameters

### Reg

The literal word `Reg` (case-insensitive). This cannot be a variable or expression.

### KeyName

The full name of the registry key.

This must start with `HKEY_LOCAL_MACHINE`, `HKEY_USERS`, `HKEY_CURRENT_USER`, `HKEY_CLASSES_ROOT`, or `HKEY_CURRENT_CONFIG` (or the abbreviations for each of these, such as `HKLM`). To access a [remote registry](#), prepend the computer name and a slash, as in this example:

```
\\workstation01\HKEY_LOCAL_MACHINE
```

### Mode

Zero or more of the following letters:

`K`: Include keys.

- V**: Include values. Values are also included if both K and V are omitted.
- R**: Recurse into subkeys. If R is omitted, keys and values within subkeys of *Key* are not included.

## Remarks

A registry-loop is useful when you want to operate on a collection registry values or subkeys, one at a time. The values and subkeys are retrieved in reverse order (bottom to top) so that [RegDelete](#) can be used inside the loop without disrupting the loop.

The following variables exist within any registry-loop. If an inner registry-loop is enclosed by an outer registry-loop, the innermost loop's registry item will take precedence:

|               |                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A_LoopRegName | Name of the currently retrieved item, which can be either a value name or the name of a subkey. Value names displayed by Windows RegEdit as "(Default)" will be retrieved if a value has been assigned to them, but A_LoopRegName will be blank for them.                                                                                                                                            |
| A_LoopRegType | The type of the currently retrieved item, which is one of the following words: KEY (i.e. the currently retrieved item is a subkey not a value), REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ, REG_DWORD, REG_QWORD, REG_BINARY, REG_LINK, REG_RESOURCE_LIST, REG_FULL_RESOURCE_DESCRIPTOR, REG_RESOURCE_REQUIREMENTS_LIST, REG_DWORD_BIG_ENDIAN (probably rare on most Windows hardware). It will be empty if |

|                       |                                                                                                                                                                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | the currently retrieved item is of an unknown type.                                                                                                                                                                                     |
| A_LoopRegKey          | The full name of the key which contains the current loop item. For remote registry access, this value will <b>not</b> include the computer name.                                                                                        |
| A_LoopRegTimeModified | The time the current subkey or any of its values was last modified. Format <code>YYYYMMDDHH24MISS</code> . This variable will be empty if the currently retrieved item is not a subkey (i.e. <i>A_LoopRegType</i> is not the word KEY). |

When used inside a registry-loop, the following commands can be used in a simplified way to indicate that the currently retrieved item should be operated upon:

|                                 |                                                                                                                                                                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RegRead, OutputVar</code> | Reads the current item. If the current item is a key, <code>ErrorLevel</code> will be set to 1 and <code>OutputVar</code> will be made empty.                                                                                   |
| <code>RegWrite [, Value]</code> | Writes to the current item. If <code>Value</code> is omitted, the item will be made 0 or blank depending on its type. If the current item is a key, <code>ErrorLevel</code> will be set to 1 and there will be no other effect. |
| <code>RegDelete</code>          | Deletes the current item. If the current item is a key, it will be deleted along with any subkeys and values it contains.                                                                                                       |

When accessing a remote registry (via the *KeyName* parameter described above), the following notes apply:

- The target machine must be running the Remote Registry service.
- Access to a remote registry may fail if the target computer is not in the same domain as yours or the local or remote username lacks sufficient

permissions (however, see below for possible workarounds).

- Depending on your username's domain, workgroup, and/or permissions, you may have to connect to a shared device, such as by mapping a drive, prior to attempting remote registry access. Making such a connection -- using a remote username and password that has permission to access or edit the registry -- may as a side-effect enable remote registry access.
- If you're already connected to the target computer as a different user (for example, a mapped drive via user Guest), you may have to terminate that connection to allow the remote registry feature to reconnect and re-authenticate you as your own currently logged-on username.
- For Windows Server 2003 and Windows XP Professional, MSDN states: "If the [registry server] computer is joined to a workgroup and the *Force network logons using local accounts to authenticate as Guest* policy is enabled, the function fails. Note that this policy is enabled by default if the computer is joined to a workgroup."
- For Windows XP Home Edition, MSDN states that "this function always fails". Home Edition lacks both the registry server and client, though the client can be extracted from one of the OS cab files.

See [Loop](#) for information about [Blocks](#), [Break](#), [Continue](#), and the `A_Index` variable (which exists in every type of loop).

## Related

[Loop](#), [Break](#), [Continue](#), [Blocks](#), [RegRead](#), [RegWrite](#), [RegDelete](#), [SetRegView](#)

## Examples

```
; Example: Delete Internet Explorer's history of
URLs typed by the user:
```

```
Loop Reg,
"HKEY_CURRENT_USER\Software\Microsoft\Internet
Explorer\TypedURLs"
 RegDelete
```

```
; Example: A working test script:
```

```
Loop Reg, "HKCU\Software\Microsoft\Windows", "R
KV" ; Recursively retrieve keys and values.
{
 if a_LoopRegType = "key"
 value := ""
 else
 {
 value := RegRead()
 if ErrorLevel
 value := "*error*"
 }
 Result := MsgBox(a_LoopRegName " = " value "
(" a_LoopRegType ")`n`nContinue?",,, "y/n")
}
Until Result = "No"
```

```
; Example: A working example to recursively search
the entire
; registry for particular value(s).
```

```

RegSearch("Notepad")
return

RegSearch(Target)
{
 ContinueRegSearch := true
 Loop Reg, "HKEY_LOCAL_MACHINE", "KVR"
 {
 Gosub, CheckThisRegItem
 if !ContinueRegSearch ; It told us to
stop.
 return
 }
 Loop Reg, "HKEY_USERS", "KVR"
 {
 Gosub, CheckThisRegItem
 if !ContinueRegSearch ; It told us to
stop.
 return
 }
 Loop Reg, "HKEY_CURRENT_CONFIG", "KVR"
 {
 Gosub, CheckThisRegItem
 if !ContinueRegSearch ; It told us to
stop.
 return
 }
 ; Note: I believe HKEY_CURRENT_USER does not
need to be searched if HKEY_USERS is
; being searched. Similarly,
HKEY_CLASSES_ROOT provides a combined view of keys
from
; HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER, so
searching all three isn't necessary.
 return
}

CheckThisRegItem:

```

```
 if A_LoopRegType = "KEY" ; Remove these two
lines if you want to check key names too.
 return
 RegValue := RegRead()
 if ErrorLevel
 return
 if InStr(RegValue, Target)
 {
 Result := MsgBox("The following match was
found:`n" A_LoopRegKey "\" A_LoopRegName "`nValue
= " RegValue "`n`nContinue?",, "y/n")
 if Result = "No"
 ContinueRegSearch := false ; Tell our
caller to stop searching.
 }
 return
}
return
```

# OnExit

Specifies a [function](#) to run automatically when the script exits.

```
OnExit Func [, AddRemove]
```

## Parameters

### Func

A function name or [function object](#) to call when the script is exiting. The function can optionally define parameters as shown below. If an OnExit function returns a non-zero integer, the script does not exit. Otherwise, the script exits after all registered functions are called.

```
MyFunction(ExitReason, ExitCode)
```

### AddRemove

One of the following values:

- 1 (the default): Call the function after any previously registered functions.
- 1: Call the function before any previously registered functions.
- 0: Do not call the function.

## Remarks

Any number of OnExit functions can be registered. An OnExit function usually should not call `ExitApp`; if it does, the script terminates immediately.

OnExit functions are called when the script exits by any means (except when it is killed by something like "End Task"). It is also called whenever the `#SingleInstance` and `Reload` commands ask a previous instance to terminate.

A script can detect and optionally abort a system shutdown or logoff via `OnMessage(0x11, "WM_QUERYENDSESSION")`.

The OnExit thread does not obey `#MaxThreads` (it will always launch when needed). In addition, while it is running, it cannot be interrupted by any thread, including hotkeys, custom menu items, and timed subroutines. However, it will be interrupted (and the script will terminate) if the user chooses Exit from the tray menu or main menu, or the script is asked to terminate as a result of `Reload` or `#SingleInstance`. Because of this, an OnExit function should be designed to finish quickly unless the user is aware of what it is doing.

If the OnExit thread encounters a failure condition such as a runtime error, the script will terminate.

If the OnExit thread was launched due to an `Exit` or `ExitApp` command that specified an exit code, that exit code is used unless an OnExit function returns true (preventing exit) or calls `ExitApp`.

Whenever an exit attempt is made, each OnExit function starts off fresh with the default values for settings such as `SendMode`. These defaults can be changed in the auto-execute section.

If an OnExit function declares parameters, its first parameter is one of the following words:

|          |                                                                                                                                                                                                                                                                                                                               |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Logoff   | The user is logging off.                                                                                                                                                                                                                                                                                                      |
| Shutdown | The system is being shut down or restarted, such as by the <a href="#">Shutdown</a> command.                                                                                                                                                                                                                                  |
| Close    | The script was sent a WM_CLOSE or WM_QUIT message, had a critical error, or is being closed in some other way. Although all of these are unusual, WM_CLOSE might be caused by <a href="#">WinClose</a> having been used on the script's main window. To prevent this, dismiss the main window with <code>Send, !{F4}</code> . |
| Error    | A runtime error occurred in a script that is not <a href="#">persistent</a> . An example of a runtime error is <a href="#">Run/RunWait</a> being unable to launch the specified program or document.                                                                                                                          |
| Menu     | The user selected Exit from the main window's menu or from the standard tray menu.                                                                                                                                                                                                                                            |
| Exit     | The <a href="#">Exit</a> or <a href="#">ExitApp</a> command was used (includes <a href="#">custom menu items</a> ).                                                                                                                                                                                                           |
| Reload   | The script is being reloaded via the <a href="#">Reload</a> command or menu item.                                                                                                                                                                                                                                             |
| Single   | The script is being replaced by a new instance of itself as a result of <a href="#">#SingleInstance</a> .                                                                                                                                                                                                                     |

## Related

[OnMessage](#), [RegisterCallback](#), [OnClipboardChange](#), [ExitApp](#), [Shutdown](#), [#Persistent](#), [Threads](#), [Gosub](#), [Return](#), [Menu](#)

## Examples

The following example uses [#Persistent](#) to prevent the script from exiting automatically. After running the script, right click the tray icon and click *Exit* to test the [OnExit](#) function. Then click "Yes" to terminate the script or "No" to keep

it running.

```
#Persistent

; Register a function to be called on exit:
OnExit("ExitFunc")

; Register an object to be called on exit:
OnExit(ObjBindMethod(MyObject, "Exiting"))

ExitFunc(ExitReason, ExitCode)
{
 if ExitReason != "Logoff" and ExitReason !=
 "Shutdown"
 {
 Result := MsgBox("Are you sure you want to
 exit?",, 4)
 if Result = "No"
 return 1 ; OnExit functions must
 return non-zero to prevent exit.
 }
 ; Do not call ExitApp -- that would prevent
 other OnExit functions from being called.
}

class MyObject
{
 Exiting()
 {
 MsgBox, MyObject is cleaning up prior to
 exiting...
 /*
 this.SayGoodbye()
 this.CloseNetworkConnections()
 */
 }
}
```



# Pause

Pauses the script's [current thread](#).

```
#p::Pause ; Pressing Win+P once will pause the script. Pressing it again will unpause.
Pause ["On|Off|Toggle", OperateOnUnderlyingThread?]
```

## Parameters

### On|Off|Toggle

If blank or omitted, it defaults to `Toggle`. Otherwise, specify one of the following words:

**Toggle** or `-1`: Pauses the [current thread](#) unless the thread beneath it is paused, in which case the underlying thread is unpause.

**On** or `1` (`true`): Pauses the current thread.

**Off** or `0` (`false`): If the thread beneath the current thread is paused, it will be in an unpaused state when resumed. Otherwise, the command has no effect.

### OperateOnUnderlyingThread?

This parameter is ignored for `Pause Off` because that always operates on the underlying thread. For the others, it is ignored unless `Pause` is being turned on (including via `Toggle`).

Specify one of the following numbers:

**0** (or omitted): The command pauses the current thread; that is, the one now running the Pause command.

**1**: The command marks the thread beneath the current thread as paused so that when it resumes, it will finish the command it was running (if any) and then enter a paused state. If there is no thread beneath the current thread, the script itself is paused, which prevents [timers](#) from running (this effect is the same as having used the menu item "Pause Script" while the script has no threads).

Note: [A\\_IsPaused](#) contains the pause state of the underlying thread.

## Remarks

Unlike [Suspend --](#) which disables [hotkeys](#) and [hotstrings](#) -- turning on pause will freeze the [current thread](#). As a side-effect, any interrupted threads beneath it will lie dormant.

Whenever any thread is paused, [timers](#) will not run. By contrast, explicitly launched threads such as [hotkeys](#) and [menu items](#) can still be launched; but when their [threads](#) finish, the underlying thread will still be paused. In other words, each thread can be paused independently of the others.

The color of the tray icon changes from green to red whenever the script's [current thread](#) is in a paused state. This color change can be avoided by freezing the icon, which is achieved by specifying 1 for the last parameter of the Menu

command. For example:

```
Menu, Tray, Icon, C:\My Icon.ico, , 1
```

To disable [timers](#) without pausing the script, use [Thread, NoTimers](#).

The Pause command is similar in function to the built-in menu item "Pause Script".

A script is always halted (though not officially paused) while it is displaying any kind of [menu](#) (tray menu, menu bar, GUI context menu, etc.)

## Related

[Suspend](#), [Menu](#), [ExitApp](#), [Threads](#), [SetTimer](#)

## Examples

```
Pause::Pause ; Assign the toggle-pause function
to the "pause" key...
#p::Pause ; ... or assign it to Win+p or some
other hotkey.
```

```
; Send a Pause command to another script.
DetectHiddenWindows, On
WM_COMMAND := 0x111
ID_FILE_PAUSE := 65403
PostMessage, %WM_COMMAND%, %ID_FILE_PAUSE%, , ,
C:\YourScript.ahk ahk_class AutoHotkey
```

# Reload

Replaces the currently running instance of the script with a new one.

## Reload

This command is useful for scripts that are frequently changed. By assigning a hotkey to this command, you can easily restart the script after saving your changes in an editor.

Any [command-line parameters](#) passed to the original script are not passed to the new instance. To pass such parameters, do not use Reload. Instead, use [Run](#) in conjunction with [A\\_AbkPath](#) and [A\\_ScriptFullPath](#) (and [A\\_IsCompiled](#) if the script is ever used in compiled form). Also, include the string `/restart` as the first parameter (i.e. after the name of the executable), which tells the program to use the same behavior as Reload. See also: [command line switches and syntax](#).

When the script restarts, it is launched in its original working directory (the one that was in effect when it was first launched). In other words, [SetWorkingDir](#) will not change the working directory that will be used for the new instance.

If the script cannot be reloaded -- perhaps because it has a syntax error -- the original instance of the script will continue running. Therefore, the reload command should be followed by whatever actions you want taken in the event of a failure (such as a [return](#) to exit the current subroutine). To have the original instance detect the failure, follow this example:

```
Reload
Sleep 1000 ; If successful, the reload will
close this instance during the Sleep, so the
line below will never be reached.
Result := MsgBox("The script could not be
reloaded. Would you like to open it for
editing?",, 4)
if Result = "Yes"
 Edit
return
```

## Related

[Edit](#)

## Example

```
^!r::Reload ; Assign Ctrl-Alt-R as a hotkey to
restart the script.
```

# Return

Returns from a subroutine to which execution had previously jumped via [function-call](#), [Gosub](#), [Hotkey](#) activation, [GroupActivate](#), or other means.

```
Return [Expression]
```

## Parameters

### Expression

This parameter should be omitted except when `return` is used inside a [function](#).

Since this parameter is an [expression](#), all of the following are valid examples:

```
return 3
return "literal string"
return MyVar
return i + 1
return true ; Returns the number 1 to mean
"true".
return itemCount < MaxItems ; Returns a
true or false value.
return FindColor(TargetColor)
```

## Remarks

The space or comma after `Return` is optional if the expression is enclosed in

parentheses, as in `return(expression)`. However, *Return* is not a function and cannot be used mid-expression.

If there is no caller to which to return, *Return* will do an [Exit](#) instead.

There are various ways to return multiple values from function to caller described within [Returning Values to Caller](#).

## Related

[Functions](#), [Gosub](#), [Exit](#), [ExitApp](#), [GroupActivate](#)

## Example

```
#Z::
MsgBox The Win-Z hotkey was pressed.
Gosub MySubroutine
return

MySubroutine:
Sleep 1000
return
```

# SetTimer

Causes a subroutine to be launched automatically and repeatedly at a specified time interval.

**SetTimer** [Label, Period|On|Off, Priority]

|                          |                          |
|--------------------------|--------------------------|
| <b>Command Example:</b>  | SetTimer "MyTimer", 100  |
| <b>Function Example:</b> | SetTimer("MyTimer", 100) |

## Parameters

### Label

The name of the [label](#) or [hotkey label](#) to which to jump, which causes the commands beneath *Label* to be executed until a [Return](#) or [Exit](#) is encountered. As with the parameters of almost all other commands, *Label* can be a [variable](#) reference such as %MyLabel%, in which case the name stored in the variable is used as the target.

If *Label* is omitted, [A\\_ThisLabel](#) will be used. For example,

`SetTimer,, Off` can be used inside a timer subroutine to turn off the timer, while `SetTimer,, 1000` would either update the current timer's *Period* or set a new timer using the label which is currently running. If [A\\_ThisLabel](#) is empty but the current thread was launched by a timer, that timer is used. This is useful for timers which launch functions or function objects.

If not a valid label name, this parameter can be the name of a function, or a single variable reference containing a [function object](#). For example, `SetTimer %funcobj%, 1000` or `SetTimer % funcobj, 1000`. Other expressions which return objects are currently unsupported. See [the class example](#) for more details.

**Note:** Passing an empty variable or an expression which results in an empty value is considered an error. This parameter must be either given a non-empty value or completely omitted.

### Period|On|Off|Delete

**On:** Re-enables a previously disabled timer at its former *period*. If the timer doesn't exist, it is created (with a default period of 250). The timer is also [reset](#). If the timer exists but was previously set to [run-only-once mode](#), it will again run only once.

**Off:** Disables an existing timer.

**Delete:** Disables and deletes an existing timer. If the timer is associated with a [function object](#), the object is released. Turning off a timer does not release the object.

**Period:** Creates or updates a timer using the absolute value of this parameter as the [approximate](#) number of milliseconds that must pass before the timer is executed. The timer will be automatically enabled and [reset](#). It can be set to repeat automatically or run only once:

- If *Period* is positive, the timer will automatically repeat until it is

explicitly disabled by the script.

- If *Period* is negative, the timer will run only once. For example, specifying -100 would run the timer 100 ms from now then disable the timer as though `SetTimer, Label, Off` had been used.

If *Label* is an object created by the script (not an actual function or label), the timer is automatically deleted after the timer function returns, unless the timer was re-enabled. This allows the object to be freed if the script is no longer referencing it, but it also means the timer's *Period* and *Priority* are not retained.

*Period* must be an integer, unless a variable or expression is used, in which case any fractional part is ignored. Its absolute value must be no larger than 4294967295 ms (49.7 days).

**Default:** If this parameter is blank and:

- 1) the timer does not exist: it will be created with a period of 250.
- 2) the timer already exists: it will be enabled and [reset](#) at its former *period* unless a *Priority* is specified.

## Priority

This optional parameter is an integer between -2147483648 and 2147483647 (or an [expression](#)) to indicate this timer's thread priority. If omitted, 0 will be used. See [Threads](#) for details.

To change the priority of an existing timer without affecting it in any other way, leave the parameter before this one blank.

## Remarks

Timers are useful because they run asynchronously, meaning that they will run at the specified frequency (interval) even when the script is waiting for a window, displaying a dialog, or busy with another task. Examples of their many uses include taking some action when the user becomes idle (as reflected by [A\\_TimeIdle](#)) or closing unwanted windows the moment they appear.

Although timers may give the illusion that the script is performing more than one task simultaneously, this is not the case. Instead, timed subroutines are treated just like other threads: they can interrupt or be interrupted by another thread, such as a [hotkey subroutine](#). See [Threads](#) for details.

Whenever a timer is created, re-enabled, or updated with a new *period*, its subroutine will not run right away; its time *period* must expire first. If you wish the timer's first execution to be immediate, use [Gosub](#) to execute the timer's subroutine (however, this will not start a new thread like the timer itself does; so settings such as [SendMode](#) will not start off at their defaults).

**Reset:** If [SetTimer](#) is used on an existing timer and parameter #2 is a number or the word ON (or it is omitted), the timer is reset; in other words, the entirety of its period must elapse before its subroutine will run again.

**Timer precision:** Due to the granularity of the OS's time-keeping system, *Period* is typically rounded up to the nearest multiple of 10 or 15.6 milliseconds (depending on the type of hardware and drivers installed). For example, a *Period* between 1 and 10 (inclusive) is usually equivalent to 10 or 15.6 on Windows

2000/XP. A shorter delay may be achieved via Loop+Sleep as demonstrated at [DllCall+timeBeginPeriod+Sleep](#).

**Reliability:** A timer might not be able to run as often as specified under the following conditions:

1. Other applications are putting a heavy load on the CPU.
2. The timer subroutine itself takes longer than its own period to run, or there are too many other competing timers.
3. The timer has been interrupted by another [thread](#), namely another timed subroutine, [hotkey subroutine](#), or [custom menu item](#) (this can be avoided via [Critical](#)). If this happens and the interrupting thread takes a long time to finish, the interrupted timer will be effectively disabled for the duration. However, any other timers will continue to run by interrupting the [thread](#) that interrupted the first timer.
4. The script is uninterruptible as a result of [Critical](#) or [Thread Interrupt/Priority](#). During such times, timers will not run. Later, when the script becomes interruptible again, any overdue timer will run once as soon as possible and then resume its normal schedule.

Although timers will operate when the script is [suspended](#), they will not run if the [current thread](#) has "[Thread NoTimers](#)" in effect or whenever any thread is [paused](#). In addition, they do not operate when the user is navigating through one of the script's menus (such as the tray icon menu or a menu bar).

Because timers operate by temporarily interrupting the script's current activity, their subroutines should be kept short (so that they finish quickly) whenever a

long interruption would be undesirable.

**Other remarks:** Timers that stay in effect for the duration of a script should usually be created in the [auto-execute section](#). By contrast, a temporary timer might often be disabled by its own subroutine (see examples at the bottom of this page).

Whenever a timed subroutine is run, it starts off fresh with the default values for settings such as [SendMode](#). These defaults can be changed in the [auto-execute section](#).

If [hotkey](#) response time is crucial (such as in games) and the script contains any timers whose subroutines take longer than about 5 ms to execute, use the following command to avoid any chance of a 15 ms delay. Such a delay would otherwise happen if a hotkey is pressed at the exact moment a timer thread is in its period of uninterruptibility:

```
Thread, interrupt, 0 ; Make all threads
always-interruptible.
```

If a timer is disabled while its subroutine is currently running, that subroutine will continue until it completes.

The [KeyHistory](#) feature shows how many timers exist and how many are currently enabled.

## Related

Gosub, Return, Threads, Thread (command), Critical, IsLabel, Menu

## Examples

```
; Example #1: Close unwanted windows whenever they appear:
```

```
SetTimer, CloseMailWarnings, 250
return
```

```
CloseMailWarnings:
```

```
WinClose, Microsoft Outlook, A timeout occurred
while communicating
```

```
WinClose, Microsoft Outlook, A connection to the
server could not be established
```

```
return
```

```
; Example #2: Wait for a certain window to appear
and then alert the user:
```

```
SetTimer, Alert1, 500
return
```

```
Alert1:
```

```
if !WinExist("Video Conversion", "Process
Complete")
```

```
 return
```

```
; Otherwise:
```

```
SetTimer,, Off ; i.e. the timer turns itself off
here.
```

```
MsgBox, The video conversion is finished.
```

```
return
```

```
; Example #3: Detection of single, double, and
triple-presses of a hotkey. This
; allows a hotkey to perform a different operation
depending on how many times
; you press it:
```

```
#C:
if winc_presses > 0 ; SetTimer already started, so
we log the keypress instead.
```

```
{
 winc_presses += 1
 return
}
```

```
; Otherwise, this is the first press of a new
series. Set count to 1 and start
; the timer:
```

```
winc_presses := 1
SetTimer, KeyWinC, -400 ; Wait for more presses
within a 400 millisecond window.
return
```

```
KeyWinC:
```

```
if winc_presses = 1 ; The key was pressed once.
{
 Run, m:\ ; Open a folder.
}
```

```
else if winc_presses = 2 ; The key was pressed
twice.
```

```
{
 Run, m:\multimedia ; Open a different folder.
}
```

```
else if winc_presses > 2
{
 MsgBox, Three or more clicks detected.
}
```

```
; Regardless of which action above was triggered,
reset the count to
; prepare for the next series of presses:
winc_presses := 0
return
```

```
; Example #4: Using a method as the timer
subroutine.
```

```
counter := new SecondCounter
counter.Start()
Sleep 5000
counter.Stop()
Sleep 2000
```

```
; An example class for counting the seconds...
```

```
class SecondCounter {
 __New() {
 this.interval := 1000
 this.count := 0
 ; Tick() has an implicit parameter "this"
 which is a reference to
 ; the object, so we need to create a
 function which encapsulates
 ; "this" and the method to call:
 this.timer := ObjBindMethod(this, "Tick")
 }
 Start() {
 ; Known limitation: SetTimer requires a
 plain variable reference.
 timer := this.timer
 SetTimer % timer, % this.interval
 Tooltip % "Counter started"
 }
}
```

```

Stop() {
 ; To turn off the timer, we must pass the
 same object as before:
 timer := this.timer
 SetTimer % timer, Off
 Tooltip % "Counter stopped at " this.count
}
; In this example, the timer calls this
method:
Tick() {
 Tooltip % ++this.count
}
}

```

#### Tips relating to the above example:

- We can also use `this.timer := this.Tick.Bind(this)`. When `this.timer` is called, it will effectively invoke `this.Tick.Call(this)` (except that `this.Tick` is not re-evaluated). By contrast, `ObjBindMethod` produces an object which invokes `this.Tick()`.
- If we rename `Tick` to `Call`, we can just use `this` directly instead of `this.timer`. This also removes the need for the temporary variable. However, `ObjBindMethod` is useful when the object has multiple methods which should be called by different event sources, such as hotkeys, menu items, GUI controls, etc.
- If the timer is being modified or deleted from within a function/method called by the timer, it may be easier to *omit the Label parameter*. In some cases this avoids the need to retain the original object which was passed to `SetTimer`, which eliminates one temporary variable (like `timer` in the

example above).

# Sleep

Waits the specified amount of time before continuing.

**Sleep** DelayInMilliseconds

|                 |                 |            |
|-----------------|-----------------|------------|
| <b>Command</b>  | <b>Example:</b> | Sleep 100  |
| <b>Function</b> | <b>Example:</b> | Sleep(100) |

## Parameters

### Delay

The amount of time to pause (in milliseconds) between 0 and 2147483647 (24 days).

## Remarks

Due to the granularity of the OS's time-keeping system, *Delay* is typically rounded up to the nearest multiple of 10 or 15.6 milliseconds (depending on the type of hardware and drivers installed). For example, a delay between 1 and 10 (inclusive) is equivalent to 10 or 15.6 on most Windows 2000/XP systems. To achieve a shorter delay, see [Examples](#).

The actual delay time might wind up being longer than what was requested if the CPU is under load. This is because the OS gives each needy process a slice of CPU time (typically 20 milliseconds) before giving another timeslice to the

script.

A delay of 0 yields the remainder of the script's current timeslice to any other processes that need it (as long as they are not significantly lower in [ProcessSetPriority](#) than the script). Thus, a delay of 0 produces an actual delay between 0 and 20ms (or more), depending on the number of needy processes (if there are no needy processes, there will be no delay at all). However, a *Delay* of 0 should always wind up being shorter than any longer *Delay* would have been.

While sleeping, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

**Sleep -1:** A delay of -1 does not sleep but instead makes the script immediately check its message queue. This can be used to force any pending [interruptions](#) to occur at a specific place rather than somewhere more random. See [Critical](#) for more details.

## Related

[SetKeyDelay](#), [SetMouseDelay](#), [SetControlDelay](#), [SetWinDelay](#)

## Examples

```
Sleep, 1000 ; 1 second
```

```
; The following is a working example that
```

demonstrates how to sleep for less time than the  
; normal 10 or 15.6 milliseconds.  
; NOTE: While a script like this is running, the  
entire operating system and all applications are  
; affected by timeBeginPeriod below.

SleepDuration := 1 ; This can sometimes be finely  
adjusted (e.g. 2 is different than 3) depending on  
the value below.

TimePeriod := 3 ; Try 7 or 3. See comment below.  
; On a PC whose sleep duration normally rounds up  
to 15.6 ms, try TimePeriod=7 to allow  
; somewhat shorter sleeps, and try TimePeriod=3 or  
less to allow the shortest possible sleeps.

DllCall("Winmm\timeBeginPeriod", uint, TimePeriod)  
; Affects all applications, not just this script's  
DllCall("Sleep"...), but does not affect SetTimer.

Iterations := 50

StartTime := A\_TickCount

Loop Iterations

    DllCall("Sleep", UInt, SleepDuration) ; Must  
use DllCall instead of the Sleep command.

DllCall("Winmm\timeEndPeriod", UInt, TimePeriod)  
; Should be called to restore system to normal.

MsgBox % "Sleep duration = " . (A\_TickCount -  
StartTime) / Iterations

# Suspend

Disables or enables all or selected [hotkeys](#) and [hotstrings](#).

**Suspend** **[Mode]**

|                 |                 |               |
|-----------------|-----------------|---------------|
| <b>Command</b>  | <b>Example:</b> | Suspend "On"  |
| <b>Function</b> | <b>Example:</b> | Suspend("On") |

## Parameters

### Mode

**On** or 1 (*true*): Suspends all [hotkeys](#) and [hotstrings](#) except those explained the Remarks section.

**Off** or 0 (*false*): Re-enables the hotkeys and hotstrings that were disable above.

**Toggle** or -1 (default): Changes to the opposite of its previous state (On or Off).

**Permit:** Does nothing except mark the current subroutine as being exempt from suspension.

## Remarks

Any hotkey/hotstring subroutine whose very first line is Suspend (except

`Suspend On`) will be exempt from suspension. In other words, the hotkey will remain enabled even while suspension is ON. This allows suspension to be turned off via such a hotkey.

The `keyboard` and/or `mouse` hooks will be installed or removed if justified by the changes made by this command.

To disable selected hotkeys or hotstrings automatically based on the type of window that is present, use `#IfWinActive/Exist`.

Suspending a script's hotkeys does not stop the script's already-running `threads` (if any); use `Pause` to do that.

When a script's hotkeys are suspended, its tray icon changes to the letter S. This can be avoided by freezing the icon, which is done by specifying 1 for the last parameter of the `Menu` command. For example:

```
Menu, Tray, Icon, C:\My Icon.ico, , 1
```

The built-in variable `A_IsSuspended` contains 1 if the script is suspended and 0 otherwise.

## Related

[#IfWinActive/Exist](#), [Pause](#), [Menu](#), [ExitApp](#)

## Example

```
^!s::Suspend ; Assign the toggle-suspend function
to a hotkey.
```

```
; Send a Suspend command to another script.
```

```
DetectHiddenWindows, On
```

```
WM_COMMAND := 0x111
```

```
ID_FILE_SUSPEND := 65404
```

```
PostMessage, %WM_COMMAND%, %ID_FILE_SUSPEND%, , ,
```

```
C:\YourScript.ahk ahk_class AutoHotkey
```

# Thread

Sets the priority or interruptibility of [threads](#). It can also temporarily disable all [timers](#).

```
Thread "NoTimers" [, false]
```

```
Thread "Priority", n
```

```
Thread "Interrupt" [, Duration, LineCount]
```

```
Command Example: Thread 100
```

```
Function Example: Thread(100)
```

**Thread "NoTimers" [, false]:** Prevents interruptions from any [timers](#) until the [current thread](#) either ends, executes `Thread "NoTimers", false`, or is interrupted by another thread that allows timers (in which case timers can interrupt the interrupting thread until it finishes).

If `Thread "NoTimers"` is not used in the auto-execute section (top part of the script), all threads start off as interruptible by timers (though the settings of `Thread "Interrupt"` [below] will still apply). By contrast, if the auto-execute section turns on *NoTimers* but never turns it off, every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timer](#)) starts off immune to interruptions by timers.

Regardless of the default setting, timers will always operate when the script has no threads (unless [Pause](#) has been turned on).

`Thread "NoTimers"` is equivalent to `Thread "NoTimers", true`. In addition, since the true/false parameter is an [expression](#), true resolves to 1, and false to 0.

**Thread "Priority", n:** Specify for **n** an integer between -2147483648 and 2147483647 (or an [expression](#)) to indicate the current thread's new priority. This has no effect on other threads. See [Threads](#) for details.

Due to its ability to buffer events, the command [Critical](#) is generally superior to `Thread "Priority"`.

On a related note, the OS's priority level for the entire script can be changed as in this example: `ProcessSetPriority, High`.

**Thread "Interrupt" [, Duration, LineCount]:** This command should be used sparingly because most scripts perform more consistently with settings close to the defaults.

By default, every newly launched thread is uninterruptible for a *Duration* of 15 milliseconds or a *LineCount* of 1000 script lines, whichever comes first. This gives the thread a chance to finish rather than being immediately interrupted by another thread that is waiting to launch (such as a buffered [hotkey](#) or a series of [timed subroutines](#) that are all due to be run).

If either component is 0, each newly launched thread is immediately interruptible. If either component is -1, the thread cannot be interrupted as a

result of that component. The maximum for both components is 2147483647.

The Interrupt setting is global, meaning that all subsequent threads will obey it, even if the setting is changed somewhere other than the [auto-execute section](#). However, [interrupted threads](#) are unaffected because their period of uninterruptibility has already expired. Similarly, the [current thread](#) is unaffected except if it is uninterruptible at the time the *LineCount* component is changed, in which case the new *LineCount* will be in effect for it.

If a [hotkey](#) is pressed or a [custom menu item](#) is selected while the [current thread](#) is uninterruptible, that event will be buffered. In other words, it will launch when the current thread finishes or becomes interruptible, whichever comes first. The exception to this is when the current thread becomes interruptible before it finishes, and it is of higher [priority](#) than the buffered event; in this case the buffered event is unbuffered and discarded.

Regardless of this setting, a thread will become interruptible the moment it displays a [MsgBox](#), [InputBox](#), [FileSelect](#), or [DirSelect](#) dialog.

Either parameter can be left blank to avoid changing it.

## Remarks

Due to its greater flexibility and its ability to buffer events, the command [Critical](#) is generally more useful than `Thread "Interrupt"` and `Thread "Priority"`.

## Related

Critical, Threads, Hotkey, Menu, SetTimer, ProcessExist

## Example

```
Thread, priority, 1 ; Make priority of current
thread slightly above average.
Thread, interrupt, 0 ; Make each newly launched
thread immediately interruptible:
Thread, interrupt, 50, 2000 ; Make each thread
interruptible after 50ms or 2000 lines, whichever
comes first.
```

# Throw

Signals the occurrence of an error. This signal can be caught by a [try-catch](#) statement.

**Throw** `[Expression]`

**Command Example:** `Throw "Error"`

## Parameters

### Expression

A value to store in [catch](#)'s `OutputVar`.

Since this parameter is an [expression](#), all of the following are valid examples:

```
throw 3
throw "literal string"
throw MyVar
throw i + 1
throw { what: "Custom error", file:
A_LineFile, line: A_LineNumber } ; Throws
an object
```

This parameter is always an expression, so variable references should not be enclosed in [percent signs](#) except to perform a [double-deref](#).

If omitted, an [exception object](#) is thrown with a default message.

## Exception(Message [, What, Extra])

Creates an object with the following properties, also common to exceptions created by [runtime errors](#):

- **Message:** An error message or [ErrorLevel](#) value.
- **What:** The name of the command, function or label which was executing or about to execute when the error occurred.
- **Extra:** Additional information about the error, if available.
- **File:** Set automatically to the full path of the script file which contains the line at which the error occurred.
- **Line:** Set automatically to the line number at which the error occurred.

If *What* is omitted, it defaults to the name of the current function or subroutine. Otherwise it can be a string or a negative offset from the top of the call stack. For example, a value of -1 sets `Exception.What` to the current function or subroutine and `Exception.Line` to the line which called it. However, if the script is [compiled](#) or the offset is invalid, *What* is simply converted to a string.

*Message* and *Extra* are converted to strings. These are displayed by an error dialog if the exception is thrown and not caught.

```
try
 BadlyCodedFunc()
catch e
 MsgBox("Error in " e.What " , which was
called at line " e.Line)
```

```
BadlyCodedFunc() {
 throw Exception("Fail", -1)
}
```

## Related

[Try](#), [Catch](#), [Finally](#)

## Examples

See [Try](#).

# Try

Guards one or more statements (commands or expressions) against runtime errors and exceptions thrown by the `throw` command.

```
Try Statement
```

```
Try
{
 Statements
}
```

## Remarks

The `try` command is usually followed by a `block` - one or more statements (commands or expressions) enclosed in braces. If only a single statement is to be executed, it can be placed on the same line as `try` or on the next line, and the braces can be omitted. To specify code that executes only when `try` catches an error, use the `catch` command.

An exception can be thrown by the `throw` command or by the program when a runtime error occurs. When an exception is thrown from within a try block or a function called by one, the following occurs:

- If there is a corresponding `catch` statement, execution continues there.
- If there is no catch statement but there is a `finally` statement, it is executed, but once it finishes the exception is automatically thrown again.

- If there is neither a catch statement nor a finally statement, execution continues at the next line outside the try block.

If an exception is thrown while no try blocks are executing, an error message is shown and the current thread exits.

The [One True Brace \(OTB\)](#) style may optionally be used with the `try` command. For example:

```
try {
 ...
} catch e {
 ...
}
```

## Related

[Catch, Throw, Finally, Blocks](#)

## Examples

```
; Example #1: The basic concept of
try/catch/throw.
```

```
try ; Attempts to execute code.
{
 HelloWorld()
 MakeToast()
}
```

```
catch e ; Handles the first error/exception
raised by the block above.
```

```

{
 MsgBox, An exception was
thrown!`nSpecifically: %e%
 Exit
}

HelloWorld() ; Always succeeds.
{
 MsgBox("Hello, world!")
}

MakeToast() ; Always fails.
{
 ; Jump immediately to the try block's error
handler:
 throw A_ThisFunc " is not implemented, sorry"
}

```

**; Example #2: Using try/catch instead of ErrorLevel.**

```

try
{
 ; The following tries to back up certain types
of files:
 FileCopy, %A_MyDocuments%*.txt,
D:\Backup\Text documents
 FileCopy, %A_MyDocuments%*.doc,
D:\Backup\Text documents
 FileCopy, %A_MyDocuments%*.jpg,
D:\Backup\Photos
}
catch
{
 MsgBox("There was a problem while backing the
files up!",, 16)
}

```

```
ExitApp
```

```
}
```

```
; Example #3: Dealing with COM errors.
```

```
try
{
 obj := ComObjCreate("ScriptControl")
 obj.ExecuteStatement('MsgBox "This is embedded
VBScript"')
 obj.InvalidMethod() ; This line produces a
runtime error.
}
catch e
{
 ; For more detail about the object that e
contains, see Exception.
 MsgBox("Exception thrown!\n\nwhat: " e.what
"\nfile: " e.file
. "\nline: " e.line "\nmessage: "
e.message "\nextra: " e.extra,, 16)
}
```

```
; Example #4: Nesting try-catch statements.
```

```
try Example1() ; Any single statement can be on
the same line with a Try command.
catch e
 MsgBox, Example1() threw %e%.

Example1()
{
 try Example2()
 catch e
 {
```

```
 if e = 1
 throw e ; Rethrow the exception so
that the caller can catch it.
 else
 MsgBox, Example2() threw %e%.
 }
}
```

```
Example2()
{
 Random, o, 1, 2
 throw o
}
```

# Until

Applies a condition to the continuation of a Loop or For-loop.

```
Loop {
 ...
} Until Expression
```

## Parameters

### Expression

Any valid [expression](#).

## Remarks

The space or comma after `Until` is optional if the expression is enclosed in parentheses, as in `until(expression)`.

The expression is evaluated once after each iteration, and is evaluated even if `continue` was used. If the expression evaluates to false (which is an empty string or the number 0), the loop continues; otherwise, the loop is broken and execution continues at the line following *Until*.

Loop Until is shorthand for the following:

```
Loop {
 ...
 if (Expression)
```

```
 break
 }
```

However, Loop Until is often easier to understand and unlike the above, can be used with a single-line action. For example:

```
Loop
 x *= 2
Until x > y
```

*Until* can be used with any Loop or For. For example:

```
Loop, Read, %A_ScriptFullPath%
 lines .= A_LoopReadLine . "`n"
Until A_Index=5 ; Read the first five lines.
MsgBox % lines
```

If *A\_Index* is used in *Expression*, it contains the index of the iteration which has just finished.

## Related

[Loop](#), [While-loop](#), [For-loop](#), [Break](#), [Continue](#), [Blocks](#), [Files-and-folders loop](#), [Registry loop](#), [File-reading loop](#), [Parsing loop](#), [If \(expression\)](#)

# While-loop

Performs a series of commands repeatedly until the specified [expression](#) evaluates to false.

```
while Expression
```

## Parameters

### Expression

Any valid [expression](#). For example: `while x < y`.

## Remarks

The expression is evaluated once before each iteration. If the expression evaluates to true (which is any result other than an empty string or the number 0), the body of the loop is executed; otherwise, execution jumps to the line following the loop's body.

A while-loop is usually followed by a [block](#), which is a collection of statements that form the *body* of the loop. However, a loop with only a single statement does not require a block (an "if" and its "else" count as a single statement for this purpose).

The One True Brace (OTB) style may optionally be used, which allows the open-brace to appear on the same line rather than underneath. For example:

```
while x < y {
```

The built-in variable **A\_Index** contains the number of the current loop iteration. It contains 1 the first time the loop's expression and body are executed. For the second time, it contains 2; and so on. If an inner loop is enclosed by an outer loop, the inner loop takes precedence. A\_Index works inside all types of loops, but contains 0 outside of a loop.

As with all loops, **Break** may be used to exit the loop prematurely. Also, **Continue** may be used to skip the rest of the current iteration, at which time A\_Index is increased by 1 and the while-loop's expression is re-evaluated. If it is still true, a new iteration begins; otherwise, the loop ends.

Specialized loops: Loops can be used to automatically retrieve files, folders, or registry items (one at a time). See [file-loop](#) and [registry-loop](#) for details. In addition, [file-reading loops](#) can operate on the entire contents of a file, one line at a time. Finally, [parsing loops](#) can operate on the individual fields contained inside a delimited string.

## Related

[Until](#), [Break](#), [Continue](#), [Blocks](#), [Loop](#), [For-loop](#), [Files-and-folders loop](#), [Registry loop](#), [File-reading loop](#), [Parsing loop](#), [If \(expression\)](#)

## Examples

```
; As the user drags the left mouse button, a
ToolTip displays the size of the region inside the
drag-area.
```

```
CoordMode, Mouse, Screen
```

```
~LButton::
```

```
 MouseGetPos, begin_x, begin_y
```

```
 while GetKeyState("LButton")
```

```
 {
```

```
 MouseGetPos, x, y
```

```
 ToolTip, % begin_x ", " begin_y "`n"
```

```
Abs(begin_x-x) " x " Abs(begin_y-y)
```

```
 Sleep, 10
```

```
 }
```

```
 ToolTip
```

```
return
```

# GUI Control Types

## Table of Contents

- Text, Edit, UpDown, Picture
- Button, Checkbox, Radio
- DropDownList, ComboBox
- ListBox, ListView, TreeView
- Link, Hotkey, DateTime
- MonthCal, Slider, Progress
- GroupBox, Tab3, StatusBar
- ActiveX (e.g. Internet Explorer Control)
- Custom

## Text

Description: A region containing borderless text that the user cannot edit. Often used to label other controls. Example:

```
Gui.Add("Text",, "Please enter your name:")
```

In this case, the last parameter is the string to display. It may contain linefeeds (`\n`) to start new lines. In addition, a single long line can be broken up into several shorter ones by means of a [continuation section](#).

If a width (W) is specified in *Options* but no [rows \(R\)](#) or height (H), the text will be word-wrapped as needed, and the control's height will be set automatically.

To detect when the user clicks the text, use the [Click event](#). For example:

```
Gui := GuiCreate()
FakeLink := Gui.Add("Text", "", "Click here to
launch Google.")
FakeLink.SetFont("underline cBlue")
FakeLink.OnEvent("Click", "LaunchGoogle")

; Alternatively, a Link control can be used:
Gui.Add("Link",, 'Click here to launch
Google.')
Gui.Show()

LaunchGoogle() {
 Run("www.google.com")
}
```

---

Text controls also support the [DoubleClick](#) event.

Only Text controls with the `SS_NOTIFY` (0x100) style send click and double-click notifications, so [OnEvent](#) automatically adds this style when a `Click` or `DoubleClick` callback is registered. On Windows Vista and later, the `SS_NOTIFY` style causes the OS to automatically copy the control's text to the clipboard when it is double-clicked.

An ampersand (&) may be used in the text to underline one of its letters. For example:

```
Gui.Add("Text",, "&First Name:")
Gui.Add("Edit")
```

In the example above, the letter F will be underlined, which allows the user to press the [shortcut key](#) `Alt+F` to set keyboard focus to the first input-capable control that was added after the text control. To instead display a literal ampersand, specify two consecutive ampersands (&&). To disable all special treatment of ampersands, include `0x80` in the control's options.

See [general options](#) for other options like *Right*, *Center*, and *Hidden*. See also: [position and sizing of controls](#).

## Edit

Description: An area where free-form text can be typed by the user. Example:

```
Gui.Add("Edit", "r9 vMyEdit", "Text to appear
inside the edit control (omit this parameter to
start off empty).")
```

The control will be multi-line if it has more than one row of text. For example, specifying `r3` in *Options* will create a 3-line edit control with the following default properties: a vertical scroll bar, word-wrapping enabled, and the Enter key captured as part of the input rather than triggering the window's [default button](#).

To start a new line in a multi-line edit control, the last parameter (contents) may contain either a solitary linefeed (``n`) or a carriage return and linefeed (``r`n`). Both methods produce literal ``r`n` pairs inside the Edit control. However, when the control's content is retrieved via `Gui.Submit` or `GuiCtrl.Value`, each ``r`n` in the text is always translated to a plain linefeed (``n`). To bypass this End-of-Line translation, use `GuiCtrl.Text`. To write the text to a file, follow this example: `FileAppend(MyEdit.Text, "C:\Saved File.txt")`.

If the control has word-wrapping enabled (which is the default for multi-line edit controls), any wrapping that occurs as the user types will not produce linefeed characters (only the Enter keystroke can do that).

Whenever the user changes the control's content, the [Change](#) event is raised.

TIP: To load a text file into an Edit control, use `FileRead` and `GuiCtrl.Value`. For example:

```
MyEdit := Gui.Add("Edit", "R20")
MyEdit.Value := FileRead("C:\My File.txt")
```

## Edit Options

To remove an option rather than adding it, precede it with a minus sign:

**Limit:** Restricts the user's input to the visible width of the edit field.

Alternatively, to limit input to a specific number of characters, include a number immediately afterward. For example, `Limit10` would allow no more than 10 characters to be entered.

**Lowercase:** The characters typed by the user are automatically converted to lowercase.

**Multi:** Makes it possible to have more than one line of text. However, it is usually not necessary to specify this because it will be auto-detected based on height (H), rows (R), or contents (*Text*).

**Number:** Prevents the user from typing anything other than digits into the field (however, it is still possible to paste non-digits into it). An alternate way of forcing a numeric entry is to attach an `UpDown` control to the Edit.

**Password:** Hides the user's input (such as for password entry) by substituting masking characters for what the user types. If a non-default masking character is

desired, include it immediately after the word Password. For example, `Password*` would make the masking character an asterisk rather than the black circle (bullet), which is the default on Windows XP. Note: This option has no effect for multi-line edit controls.

**ReadOnly:** Prevents the user from changing the control's contents. However, the text can still be scrolled, selected and copied to the clipboard.

**Tn:** The letter T may be used to set tab stops inside a [multi-line edit control](#) (since tab stops determine the column positions to which literal TAB characters will jump, they can be used to format the text into columns). If the letter T is not used, tab stops are set at every 32 dialog units (the width of each "dialog unit" is determined by the operating system). If the letter T is used only once, tab stops are set at every n units across the entire width of the control. For example, `Gui.Add("Edit", "vMyEdit r16 t64")` would double the default distance between tab stops. To have custom tab stops, specify the letter T multiple times as in the following example: `Gui.Add("Edit", "vMyEdit r16 t8 t16 t32 t64 t128")`. One tab stop is set for each of the absolute column positions in the list, up to a maximum of 50 tab stops. Note: Tab stops require a multiline edit control.

**Uppercase:** The characters typed by the user are automatically converted to uppercase.

**WantCtrlA:** Specify `-WantCtrlA` (minus `WantCtrlA`) to prevent the user's press of Control-A from selecting all text in the edit control.

**WantReturn:** Specify `-WantReturn` (that is, a minus sign followed by `WantReturn`) to prevent a multi-line edit control from capturing the Enter keystroke. Pressing Enter will then be the same as pressing the window's [default button](#) (if any). In this case, the user may press Control-Enter to start a new line.

**WantTab:** Causes a tab keystroke to produce a tab character rather than navigating to the next control. Without this option, the user may press Control-Tab to produce a tab character inside a multi-line edit control. Note: Although *WantTab* also works in a single-line edit control, each tab character is displayed as an empty-box character (though it is stored as a real tab).

**-Wrap** (minus wrap): Turns off word-wrapping in a multi-line edit control. Since this style cannot be changed after the control has been created, use one of the following to change it: 1) [Destroy](#) then recreate the window and its control; or 2) Create two overlapping edit controls, one with wrapping enabled and the other without it. The one not currently in use can be kept empty and/or hidden.

See [general options](#) for other options like *Right*, *Center*, and *Hidden*. See also: [position and sizing of controls](#).

**A more powerful edit control:** HiEdit is a free, multitabled, large-file edit control consuming very little memory. It can edit both text and binary files. For details and a demonstration, see [www.autohotkey.com/forum/topic19141.html](http://www.autohotkey.com/forum/topic19141.html)

## UpDown

Description: A pair of arrow buttons that the user can click to increase or decrease a value. By default, an UpDown control automatically snaps onto the previously added control. This previous control is known as the UpDown's *buddy control*. The most common example is a "spinner", which is an UpDown attached to an [Edit control](#). For example:

```
Gui.Add("Edit")
Gui.Add("UpDown", "vMyUpDown Range1-10", 5)
```

In the example above, the Edit control is the UpDown's buddy control. Whenever the user presses one of the arrow buttons, the number in the Edit control is automatically increased or decreased.

An UpDown's buddy control can also be a [Text control](#) or [ListBox](#). However, due to OS limitations, controls other than these (such as [ComboBox](#) and [DropDownList](#)) might not work properly with the [Change](#) event and other features.

Specify the UpDown's starting position as the last parameter (if omitted, it starts off at 0 or the number in the allowable range that is closest to 0).

When [Gui.Submit](#) or [GuiCtrl.Value](#) is used, the return value is the current numeric position of the UpDown. If the UpDown is attached to an Edit control and you do not wish to validate the user's input, it is best to use the UpDown's value rather than the Edit's. This is because the UpDown will always yield an in-

range number, even when the user has typed something non-numeric or out-of-range in the Edit control. On a related note, numbers with more than three digits get a [thousands separator](#) (such as comma) by default. These separators are returned by the Edit control but not by the UpDown control.

Whenever the user clicks one of the arrow buttons or presses an arrow key on the keyboard, the [Change](#) event is raised.

## UpDown Options

**Horz:** Make's the control's buttons point left/right rather than up/down. By default, *Horz* also makes the control isolated (no buddy). This can be overridden by specifying `Horz 16` in the control's options.

**Left:** Puts the UpDown on the left side of its buddy rather than the right.

**Range:** Sets the range to be something other than 0 to 100. After the word Range, specify the minimum, a dash, and maximum. For example, Range1-1000 would allow a number between 1 and 1000 to be selected; Range-50-50 would allow a number between -50 and 50; and Range-10--5 would allow a number between -10 and -5. The minimum and maximum may be swapped to cause the arrows to move in the opposite of their normal direction. The broadest allowable range is -2147483648-2147483647. Finally, if the buddy control is a [ListBox](#), the range defaults to 32767-0 for verticals and the inverse for horizontals ([Horz](#)).

**Wrap:** Causes the control to wrap around to the other end of its range when the user attempts to go beyond the minimum or maximum. Without *Wrap*, the

control stops when the minimum or maximum is reached.

**-16 (minus 16):** Causes a vertical UpDown to be isolated; that is, it will have no buddy. This also causes the control to obey any specified width, height, and position rather than conforming to the size of its buddy control. In addition, an isolated UpDown tracks its own position internally. This position can be retrieved normally by means such as [Gui.Submit](#).

**0x80:** Include `0x80` in *Options* to omit the thousands separator that is normally present between every three decimal digits in the buddy control. However, this style is normally not used because the separators are omitted from the number whenever the script retrieves it from the UpDown control itself (rather than its buddy control).

**Increments other than 1:** In [this script](#), NumEric demonstrates how to change an UpDown's increment to a value other than 1 (such as 5 or 0.1).

See also: [position and sizing of controls](#).

## Picture (or Pic)

Description: An area containing an image (see last two paragraphs for supported file types). The last parameter is the filename of the image, which is assumed to be in `A_WorkingDir` if an absolute path isn't specified. Example:

```
Gui.Add("Picture", "w300 h-1", "C:\My
Pictures\Company Logo.gif")
```

To retain the image's actual width and/or height, omit the `W` and/or `H` options. Otherwise, the image is scaled to the specified width and/or height (this width and height also determines which icon to load from a multi-icon `.ICO` file). To shrink or enlarge the image while preserving its aspect ratio, specify `-1` for one of the dimensions and a positive number for the other. For example, specifying `w200 h-1` would make the image 200 pixels wide and cause its height to be set automatically. If the picture cannot be loaded or displayed (e.g. file not found), the control is left empty and its width and height are set to zero.

Picture controls support the `Click` and `DoubleClick` events, with the same caveat as Text controls.

To use a picture as a background for other controls, the picture should normally be added prior to those controls. However, if those controls are input-capable and the picture has the `SS_NOTIFY` style (which may be added automatically by `OnEvent`), create the picture after the other controls and include `0x4000000` (which is `WS_CLIPSIBLINGS`) in the picture's `Options`. This trick also allows a

picture to be the background behind a [Tab control](#) or [ListView](#).

**Icons, cursors, and animated cursors:** Icons and cursors may be loaded from the following types of files: ICO, CUR, ANI, EXE, DLL, CPL, SCR, and other types that contain icon resources. To use an icon group other than the first one in the file, include in *Options* the word Icon followed by the number of the group. In the following example, the default icon from the second icon group would be used: `Gui.Add("Picture", "Icon2", "C:\My Application.exe").`

Specifying the word *AltSubmit* in *Options* tells the program to use Microsoft's GDIPlus.dll to load the image, which might result in a different appearance for GIF, BMP, and icon images. For example, it would load a GIF that has a transparent background as a transparent bitmap, which allows the [BackgroundTrans](#) option to take effect (but icons support transparency without *AltSubmit*). If GDIPlus is not available (see next paragraph), *AltSubmit* is ignored and the image is loaded using the normal method.

All operating systems support GIF, JPG, BMP, ICO, CUR, and ANI images. On Windows XP or later, additional image formats such as PNG, TIF, Exif, WMF, and EMF are supported. Operating systems older than XP can be given support by copying Microsoft's free GDI+ DLL into the AutoHotkey.exe folder (but in the case of a [compiled script](#), copy the DLL into the script's folder). To download the DLL, search for the following phrase at [www.microsoft.com](http://www.microsoft.com): gdi redistributable

**Animated GIFs:** Although animated GIF files can be displayed in a picture

control, they will not actually be animated. To solve this, use the AniGIF DLL (which is free for non-commercial use) as demonstrated at [www.autohotkey.com/forum/topic19264.html](http://www.autohotkey.com/forum/topic19264.html)

A `bitmap` or `icon handle` can be used instead of a filename. For example,

```
HBITMAP:%handle%
```

## Button

Description: A pushbutton, which can be pressed to trigger an action. In this case, the last parameter is the name of the button (shown on the button itself), which may include linefeeds (`\n`) to start new lines. Example:

```
MyBtn := Gui.Add("Button", "Default", "OK")
MyBtn.OnEvent("Click", "MyBtn_Click") ; Call
MyBtn_Click when clicked.
```

The `Click` event is raised whenever the user clicks the button or presses Space or Enter while it has the focus.

The `DoubleClick`, `Focus` and `LoseFocus` events are also supported. As these events are only raised if the control has the `BS_NOTIFY` (0x4000) style, it is added automatically by `OnEvent`.

The example above includes the word **Default** in its *Options* to make "OK" the default button. The default button's `Click` event is automatically triggered whenever the user presses ENTER, except when the keyboard focus is on a different button or a multi-line edit control having the `WantReturn` style. To later change the default button to another button, follow this example, which makes the Cancel button become the default:

```
Gui.Control["Cancel"].Opt("+Default").
```

To later change the window to have no default button, follow this example:

```
Gui.Control["OK"].Opt("-default").
```

An ampersand (&) may be used in the name button to underline one of its letters. For example:

```
Gui.Add("Button",, "&Pause")
```

In the example above, the letter P will be underlined, which allows the user to press Alt+P as [shortcut key](#). To display a literal ampersand, specify two consecutive ampersands (&&).

Known limitation: Certain desktop themes might not display a button's text properly. If this occurs, try including `-Wrap` (minus Wrap) in the button's options. However, this also prevents having more than one line of text.

## Checkbox

Description: A small box that can be checked or unchecked to represent On/Off, Yes/No, etc. Example:

```
Gui.Add("Checkbox", "vShipToBillingAddress",
"Ship to billing address?")
```

The last parameter is a label displayed next to the box, which is typically used as a prompt or description of what the checkbox does. It may include linefeeds (`\n`) to start new lines. If a width (W) is specified in *Options* but no `rows (R)` or height (H), the control's text will be word-wrapped as needed, and the control's height will be set automatically.

`GuiCtrl.Value` returns the number 1 for checked, 0 for unchecked, and -1 for gray/indeterminate.

Specify the word **Check3** in *Options* to enable a third "indeterminate" state that displays a gray checkmark or a square instead of a black checkmark (the indeterminate state indicates that the checkbox is neither checked nor unchecked). Specify the word **Checked** or **CheckedGray** in *Options* to have the checkbox start off checked or indeterminate, respectively. The word **Checked** may optionally be followed immediately by a 0, 1, or -1 to indicate the starting state. In other words, `Checked` and `Checked%VarContainingOne%` are the same.

Whenever the checkbox is clicked, it automatically cycles between its two or

three possible states, and then raises the `Click` event, allowing the script to immediately respond to the user's input.

The `DoubleClick`, `Focus` and `LoseFocus` events are also supported. As these events are only raised if the control has the `BS_NOTIFY` (0x4000) style, it is added automatically by `OnEvent`. This style is not applied by default as it prevents rapid clicks from changing the state of the checkmark (such as if the user clicks twice to toggle from unchecked to checked and then to indeterminate).

Known limitation: Certain desktop themes might not display a button's text properly. If this occurs, try including `-Wrap` (minus `Wrap`) in the button's options. However, this also prevents having more than one line of text.

## Radio

A Radio button is a small empty circle that can be checked (on) or unchecked (off). Example:

```
Gui.Add("Radio", "vMyRadioGroup", "Wait for all
items to be in stock before shipping.")
```

These controls usually appear in *radio groups*, each of which contains two or more radio buttons. When the user clicks a radio button to turn it on, any others in its radio group are turned off automatically (the user may also navigate inside a group with the arrow keys). A radio group is created automatically around all consecutively added radio buttons. To start a new group, specify the word **Group** in the *Options* of the first button of the new group -- or simply add a non-radio control in between, since that automatically starts a new group.

For the last parameter, specify the label to display to the right of the radio button. This label is typically used as a prompt or description, and it may include linefeeds (`n) to start new lines. If a width (W) is specified in *Options* but no rows (R) or height (H), the control's text will be word-wrapped as needed, and the control's height will be set automatically.

Specify the word **Checked** in *Options* to have the button start off in the "on" state. The word Checked may optionally be followed immediately by a 0 or 1 to indicate the starting state: 0 for unchecked and 1 for checked. In other words, `Checked` and `Checked%VarContainingOne%` are the same.

`GuiCtrl.Value` returns the number 1 for "on" and 0 for "off". To instead retrieve the position number of the selected radio option within a radio group, `name` only one of the radio buttons and use `Gui.Submit`.

The `Click` event is raised whenever the user turns on the button. Unlike the single-variable mode in the previous paragraph, the event callback must be registered for each button in a radio group for which it should be called. This allows the flexibility to ignore the clicks of certain buttons.

The `DoubleClick`, `Focus` and `LoseFocus` events are also supported. As these events are only raised if the control has the `BS_NOTIFY` (0x4000) style, it is added automatically by `OnEvent`.

Known limitation: Certain desktop themes might not display a button's text properly. If this occurs, try including `-wrap` (minus `Wrap`) in the button's options. However, this also prevents having more than one line of text.

## DropDownList (or DDL)

Description: A list of choices that is displayed in response to pressing a small button. In this case, the last parameter is a pipe-delimited list of choices such as `Choice1|Choice2|Choice3` or an array like `["Choice1", "Choice2", "Choice3"]`. Example:

```
Gui.Add("DropDownList", "vColorChoice",
"Black|White|Red|Green|Blue")
```

To have one of the items pre-selected when the window first appears, include two pipe characters after it (e.g. `Red|Green||Blue`). Alternatively, include in *Options* the word **Choose** followed immediately by the number to pre-select. For example, `Choose5` would pre-select the fifth item (as with other options, it can also be a variable such as `Choose%Var%`). After the control is created, use `Value`, `Text` or `Choose` to change the selection, and `Add` or `Delete` to add or remove entries from the list.

Specify either the word **Uppercase** or **Lowercase** in *Options* to automatically convert all items in the list to uppercase or lowercase. Specify the word **Sort** to automatically sort the contents of the list alphabetically (this also affects any items added later via `GuiCtrl.Add`). The `Sort` option also enables incremental searching whenever the list is dropped down; this allows an item to be selected by typing the first few characters of its name.

When `Gui.Submit` or `GuiCtrl.Value` is used, the return value is the position

number of the currently selected item (the first item is 1, the second is 2, etc.) or zero if none is selected. To get its text instead, use `GuiCtrl.Text`.

Whenever the user selects a new item, the `Change` event is raised. The `Focus` and `LoseFocus` events are also supported.

Use the `R` or `H` option to control the height of the popup list. For example, specifying `R5` would make the list 5 rows tall, while `H400` would set the total height of the selection field and list to 400 pixels. If both `R` and `H` are omitted, the list will automatically expand to take advantage of the available height of the user's desktop (however, operating systems older than Windows XP will show 3 rows by default).

To set the height of the selection field or list items, use the `CB_SETITEMHEIGHT` message as in the example below:

```
Gui := GuiCreate()
DDL := Gui.Add("DDL", "vcbx w200", "One|Two")
; CB_SETITEMHEIGHT = 0x153
PostMessage(0x153, -1, 50,, "ahk_id " DDL.Hwnd)
; Set height of selection field.
PostMessage(0x153, 0, 50,, "ahk_id " DDL.Hwnd)
; Set height of list items.
Gui.Show("h70")
```

The separator between fields may be changed to something other than pipe (`|`). For example `Gui.Opt("+Delimiter n")` would change it to linefeed and `Gui.Opt("+DelimiterTab")` would change it to tab (`^t`).

## ComboBox

Description: Same as DropDownList but also permits free-form text to be entered as an alternative to picking an item from the list. Example:

```
Gui.Add("ComboBox", "vColorChoice",
"Red|Green|Blue|Black|White")
```

`GuiCtrl.Value` returns the position number of the currently selected item (the first item is 1, the second is 2, etc.) or 0 if the control contains text which does not match a list item. To get the contents of the ComboBox's edit field, use `GuiCtrl.Text`. `Gui.Submit` stores the text, unless the word **AltSubmit** is in the control's *Options* and the text matches a list item, in which case it stores the position number of the item.

Whenever the user selects a new item or changes the control's text, the `Change` event is raised. The `Focus` and `LoseFocus` events are also supported.

## ListBox

Description: A relatively tall box containing a list of choices that can be selected. In this case, the last parameter is a pipe-delimited list of choices such as `Choice1|Choice2|Choice3` or an array like `["Choice1", "Choice2", "Choice3"]`. Example:

```
Gui.Add("ListBox", "vColorChoice",
"Red|Green|Blue|Black|White")
```

To have list item(s) pre-selected when the window first appears, include two pipe characters after each (the `Multi` option is required if more than one item is to be pre-selected). Alternatively, include in *Options* the word **Choose** followed immediately by a single item number to pre-select. For example, `Choose5` would pre-select the fifth item. After the control is created, use `Value`, `Text` or `Choose` to change the selection, and `Add` or `Delete` to add or remove entries from the list.

If the `Multi` option is absent, `GuiCtrl.Value` returns the position number of the currently selected item (the first item is 1, the second is 2, etc.) or 0 if there is no item selected. To get the selected item's text instead, use `GuiCtrl.Text`. If the `Multi` option is used, `Value` and `Text` return an array of items instead of a single item.

`Gui.Submit` stores `Text` by default, but stores `Value` instead if the word `AltSubmit` is in the control's *Options*.

Whenever the user selects or deselects one or more items, the [Change](#) event is raised. The [DoubleClick](#), [Focus](#) and [LoseFocus](#) events are also supported.

When adding a large number of items to a `ListBox`, performance may be improved by using `MyListBox.Opt("-Redraw")` prior to the operation, and `MyListBox.Opt("+Redraw")` afterward.

## ListBox Options

**Choose:** See [above](#).

**Multi:** Allows more than one item to be selected simultaneously via shift-click and control-click (to avoid the need for shift/control-click, specify [the number 8](#) instead of the word Multi). In this case, `Gui.Submit` or `GuiCtrl.Value` returns an array of selected position numbers. For example, `[1, 2, 3]` would indicate that the first three items are selected. To get an array of selected texts instead, use `GuiCtrl.Text`. To extract the individual items from the array, use `MyListBox.Text[1]` (1 would be the first item) or a [For-loop](#) such as this example:

```
For Index, Field in MyListBox.Text
{
 MsgBox Selection number %Index% is %Field%.
}
```

**ReadOnly:** Prevents items from being visibly highlighted when they are selected (but `Gui.Submit`, `GuiCtrl.Value` or `GuiCtrl.Text` will still return the selected item).

**Sort:** Automatically sorts the contents of the list alphabetically (this also affects any items added later via [GuiCtrl.Add](#)). The Sort option also enables incremental searching, which allows an item to be selected by typing the first few characters of its name.

**Tn:** The letter T may be used to set tab stops, which can be used to format the text into columns. If the letter T is not used, tab stops are set at every 32 dialog units (the width of each "dialog unit" is determined by the operating system). If the letter T is used only once, tab stops are set at every **n** units across the entire width of the control. For example, `Gui.Add("ListBox", "vMyListBox t64")` would double the default distance between tab stops. To have custom tab stops, specify the letter T multiple times as in the following example:

```
Gui.Add("ListBox", "vMyListBox t8 t16 t32 t64 t128").
```

One tab stop is set for each of the absolute column positions in the list, up to a maximum of 50 tab stops.

**0x100:** Include 0x100 in options to turn on the LBS\_NOINTEGRALHEIGHT style. This forces the ListBox to be exactly the height specified rather than a height that prevents a partial row from appearing at the bottom. This option also prevents the ListBox from shrinking when its font is changed.

To specify the number of rows of text (or the height and width), see [position and sizing of controls](#).

## ListView and TreeView

See separate pages [ListView](#) and [TreeView](#).

## Link

Description: A text control that can contain links similar to those found in a web browser. Within the control's text, enclose the link text within `<A>` and `</A>` to create a clickable link. Although this looks like HTML, Link controls only support the opening `<A>` tag (optionally with an ID and/or HREF attribute) and closing `</A>` tag. For example:

```
Gui.Add("Link",, 'This is a link')
```

The text in the example above translates to **This is a link**.

Whenever the user clicks on a link, the `Click` event is raised. If the control has no `Click` callback (registered by calling `OnEvent`), the link's HREF is automatically executed as though passed to the `Run` function.

```
Gui := GuiCreate()
Link := Gui.Add("Link",
 , 'Click to run <a href="notepad"
 id="notepad">Notepad or open '
 . '<a id="help"
 href="https://autohotkey.com/docs/">online
 help.')
Link.OnEvent("Click", "Link_Click")
Link_Click(Ctrl, ID, HREF)
{
 if MsgBox("ID: " ID "`nHREF: " HREF
 "`n`nExecute this link?",, "y/n") = "yes"
 Run(HREF)
```



## Hotkey

Description: A box that looks like a single-line edit control but instead accepts a keyboard combination pressed by the user. For example, if the user presses Control+Alt+C on an English keyboard layout, the box would display "Ctrl + Alt + C".

```
Gui.Add("Hotkey", "vChosenHotkey")
```

`GuiCtrl.Value` returns the control's hotkey modifiers and name, which are compatible with the `Hotkey` command. Examples: `^!C`, `+!Home`, `+^Down`, `^Numpad1`, `!NumpadEnd`. If there is no hotkey in the control, the value is blank. Note: Some keys are displayed the same even though they are retrieved as different names. For example, both `^Numpad7` and `^NumpadHome` might be displayed as Ctrl + Num 7.

By default, the control starts off with no hotkey specified. To instead have a default, specify its modifiers and name as the last parameter as in this example:

```
Gui.Add("Hotkey", "vChosenHotkey", "^!p")
```

The only modifiers supported are ^ (Control), ! (Alt), and + (Shift). See the [key list](#) for available key names.

Whenever the user changes the control's content (by pressing a key), the `Change` event is raised. Note: The event is raised even when an incomplete hotkey is present. For example, if the user holds down the Control key, the event is raised once and `Value` returns only a circumflex (^). When the user completes the

hotkey, the event is raised again and [Value](#) returns the complete hotkey.

To restrict the types of hotkeys the user may enter, include the word **Limit** followed by the sum of one or more of the following numbers:

1: Prevent unmodified keys

2: Prevent Shift-only keys

4: Prevent Control-only keys

8: Prevent Alt-only keys

16: Prevent Shift-Control keys

32: Prevent Shift-Alt keys

64: This value is not supported (it will not behave correctly).

128: Prevent Shift-Control-Alt keys.

For example, `Limit1` would prevent unmodified hotkeys such as letters and numbers from being entered, and `Limit15` would require at least two modifier keys. If the user types a forbidden modifier combination, the Control+Alt combination is automatically and visibly substituted.

The Hotkey control has limited capabilities. For example, it does not support mouse/joystick hotkeys or the Windows key (LWin and RWin). One way to work around this is to provide one or more [checkboxes](#) as a means for the user to enable extra modifiers such as the Windows key.

## DateTime

Description: A box that looks like a single-line edit control but instead accepts a date and/or time. A drop-down calendar is also provided. Example:

```
Gui.Add("DateTime", "vMyDateTime", "LongDate")
```

The last parameter is a format string, as described below.

### SetFormat

Sets the display format of a DateTime control.

```
DateTime.SetFormat([Format])
```

#### Format

One of the following:

**ShortDate** (or omitted/blank): Uses the locale's short date format. For example, in some locales it would look like: 6/1/2005

## DateTime Usage

To have a date other than today pre-selected, include in *Options* the word **Choose** followed immediately by a date in YYYYMMDD format. For example, `Choose20050531` would pre-select May 31, 2005 (as with other options, it

can also be a variable such as `Choose%Var%`). To have no date/time selected, specify **ChooseNone**. *ChooseNone* also creates a checkbox inside the control that is unchecked whenever the control has no date. Whenever the control has no date, `Gui.Submit` or `GuiCtrl.Value` will retrieve a blank value (empty string).

The time of day may optionally be present. However, it must always be preceded by a date when going into or coming out of the control. The format of the time portion is HH24MISS (hours, minutes, seconds), where HH24 is expressed in 24-hour format; for example, 09 is 9am and 21 is 9pm. Thus, a complete date-time string would have the format `YYYYMMDDHH24MISS`.

When specifying dates in the `YYYYMMDDHH24MISS` format, only the leading part needs to be present. Any remaining element that has been omitted will be supplied with the following default values:

MM: Month 01

DD: Day 01

HH24: Hour 00

MI: Minute 00

SS: Second 00

Within the drop-down calendar, the today-string at the bottom can be clicked to select today's date. In addition, the year and month name are clickable and allow easy navigation to a new month or year.

Keyboard navigation: Use the Up/Down arrow keys, NumpadPlus/Minus, and Home/End to increase or decrease the control's values. Use LeftArrow and RightArrow to move from field to field inside the control. Within the drop-down

calendar, use the arrow keys to move from day to day; use PageUp/Down to move backward/forward by one month; use Ctrl-PageUp/Down to move backward/forward by one year; and use Home/End to select the first/last day of the month.

When `Gui.Submit` or `GuiCtrl.Value` is used, the return value is the selected date and time in `YYYYMMDDHH24MISS` format. Both the date and the time are present regardless of whether they were actually visible in the control.

Whenever the user changes the date or time, the `Change` event is raised. The `Focus` and `LoseFocus` events are also supported.

## DateTime Options

**Choose:** See [above](#).

**Range:** Restricts how far back or forward in time the selected date can be. After the word Range, specify the minimum and maximum dates in `YYYYMMDD` format (with a dash between them). For example, `Range20050101-20050615` would restrict the date to the first 5.5 months of 2005. Either the minimum or maximum may be omitted to leave the control unrestricted in that direction. For example, `Range20010101` would prevent a date prior to 2001 from being selected and `Range-20091231` (leading dash) would prevent a date later than 2009 from being selected. Without the Range option, any date between the years 1601 and 9999 can be selected. The time of day cannot be restricted.

**Right:** Causes the drop-down calendar to drop down on the right side of the control instead of the left.

**1:** Specify the number 1 in *Options* to provide an up-down control to the right of the control to modify date-time values, which replaces the button of the drop-down month calendar that would otherwise be available. This does not work in conjunction with the format option `LongDate` described above.

**2:** Specify the number 2 in *Options* to provide a checkbox inside the control that the user may uncheck to indicate that no date/time is selected. Once the control is created, this option cannot be changed.

## MonthCal

Description: A tall and wide control that displays all the days of the month in calendar format. The user may select a single date or a range of dates. Example:

```
Gui.Add("MonthCal", "vMyCalendar")
```

To have a date other than today pre-selected, specify it as the third parameter in YYYYMMDD format (e.g. `20050531`). A range of dates may also be pre-selected by including a dash between two dates (e.g. `20050525-20050531`).

It is usually best to omit width (W) and height (H) for a MonthCal because it automatically sizes itself to fit exactly one month. To display more than one month vertically, specify `R2` or higher in *Options*. To display more than one month horizontally, specify `W-2` (W negative two) or higher. These options may both be present to expand in both directions.

The today-string at the bottom of the control can be clicked to select today's date. In addition, the year and month name are clickable and allow easy selection of a new year or month.

Unlike [DateTime](#)'s drop-down calendar, keyboard navigation is generally not supported in a MonthCal.

When [Gui.Submit](#) or [GuiCtrl.Value](#) is used, the return value is the selected date in YYYYMMDD format (without any time portion). However, when the [multi-select](#) option is in effect, the minimum and maximum dates are retrieved with a

dash between them (e.g. `20050101-20050108`). If only a single date was selected in a multi-select calendar, the minimum and maximum are both present but identical. `StrSplit` can be used to separate the dates. For example, the following would put the minimum in `Date[1]` and the maximum in `Date[2]`:

```
Date := StrSplit(MyMonthCal.Value, "-").
```

The `Change` event is raised when: 1) the user changes the selection; or 2) every two minutes in case a new day has arrived (this behavior is a quirk of the OS).

When specifying dates in the `YYYYMMDD` format, the `MM` and/or `DD` portions may be omitted, in which case they are assumed to be 1. For example, `200205` is seen as `20020501`, and `2005` is seen as `20050101`.

## MonthCal Options

**Multi:** Multi-select. Allows the user to shift-click or click-drag to select a range of adjacent dates (the user may still select a single date too). This option may be specified explicitly or put into effect automatically by means of specifying a selection range when the control is created. For example:

```
Gui.Add("MonthCal", "vMyCal", "20050101-20050108").
```

Once the control is created, this option cannot be changed.

**Range:** Restricts how far back or forward in time the calendar can go. After the word `Range`, specify the minimum and maximum dates in `YYYYMMDD` format (with a dash between them). For example, `Range20050101-20050615` would restrict the selection to the first 5.5 months of 2005. Either the minimum or maximum may be omitted to leave the calendar unrestricted in that direction.

For example, `Range20010101` would prevent a date prior to 2001 from being selected and `Range-20091231` (leading dash) would prevent a date later than 2009 from being selected. Without the Range option, any date between the years 1601 and 9999 can be selected.

**4:** Specify the number 4 in *Options* to display week numbers (1-52) to the left of each row of days. Week 1 is defined as the first week that contains at least four days.

**8:** Specify the number 8 in *Options* to prevent the circling of today's date within the control.

**16:** Specify the number 16 in *Options* to prevent the display of today's date at the bottom of the control.

## Slider

Description: A sliding bar that the user can move along a vertical or horizontal track. The standard volume control in the taskbar's tray is an example of a slider.

Example:

```
Gui.Add("Slider", "vMySlider", 50)
```

Specify the starting position of the slider as the last parameter. If the last parameter is omitted, the slider starts off at 0 or the number in the allowable range that is closest to 0.

The user may slide the control by the following means: 1) dragging the bar with the mouse; 2) clicking inside the bar's track area with the mouse; 3) turning the mouse wheel while the control has focus; or 4) pressing the following keys while the control has focus: Arrow keys, Page-up, Page-down, Home, and End.

`GuiCtrl.Value` and `Gui.Submit` return or store the current numeric position of the slider.

## Detecting Changes

By default, the slider's `Change` event is raised when the user has stopped moving the slider, such as by releasing the mouse button after having dragging it. If the control has the `AltSubmit` option, the `Change` event is also raised (very frequently) after each visible movement of the bar while the user is dragging it with the mouse.

Ctrl\_ **Change**(GuiCtrlObj, Info)

### Info

A numeric value from the table below indicating how the slider was moved. These values and the corresponding names are defined in the Windows SDK.

| Value                                        | Name             | Meaning                                                                                                                                                   |
|----------------------------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                                            | TB_LINEUP        | The user pressed the Left-arrow or Up-arrow key.                                                                                                          |
| 1                                            | TB_LINEDOWN      | The user pressed the Right-arrow or Down-arrow key.                                                                                                       |
| 2                                            | TB_PAGEUP        | The user pressed the Page-up key.                                                                                                                         |
| 3                                            | TB_PAGEDOWN      | The user pressed the Page-down key.                                                                                                                       |
| 4                                            | TB_THUMBPOSITION | The user moved the slider via the mouse wheel, or finished a drag-and-drop to a new position.                                                             |
| 6                                            | TB_TOP           | The user pressed the Home key to send the slider to the left or top side.                                                                                 |
| 7                                            | TB_BOTTOM        | The user pressed the End key to send the slider to the right or bottom side.                                                                              |
| <b>Only if the AltSubmit option is used:</b> |                  |                                                                                                                                                           |
| 5                                            | TB_THUMBTRACK    | The user is currently dragging the slider via the mouse; that is, the mouse button is currently down.                                                     |
| 8                                            | TB_ENDTRACK      | The user has finished moving the slider, either via the mouse or the keyboard. Note: With the exception of mouse wheel movement (#4), the Change event is |

raised again for #8 even though it was already raised with one of the digits above.

## Slider Options

**Buddy1** and **Buddy2**: Specifies up to two existing controls to automatically reposition at the ends of the slider. Buddy1 is displayed at the left or top side (depending on whether the Vertical option is present). Buddy2 is displayed at the right or bottom side. After the word Buddy1 or Buddy2, specify the [Name](#) or [HWND](#) of an existing control. For example, `Buddy1MyTopText` would assign the control whose name is MyTopText. The text or ClassNN of a control can also be used, but only up to the first space or tab.

**Center**: The thumb (the bar moved by the user) will be blunt on both ends rather than pointed at one end.

**Invert**: Reverses the control so that the lower value is considered to be on the right/bottom rather than the left/top. This is typically used to make a vertical slider move in the direction of a traditional volume control. Note: The ToolTip option described below will not obey the inversion and therefore should not be used in this case.

**Left**: The thumb (the bar moved by the user) will point to the top rather than the bottom. But if the Vertical option is in effect, the thumb will point to the left rather than the right.

**Line**: Specifies the number of positions to move when the user presses one of

the arrow keys. After the word **Line**, specify number of positions to move. For example: `Line2`.

**NoTicks:** Omits tickmarks alongside the track.

**Page:** Specifies the number of positions to move when the user presses the Page-up or Page-down key. After the word **Page**, specify number of positions to move. For example: `Page10`.

**Range:** Sets the range to be something other than 0 to 100. After the word **Range**, specify the minimum, a dash, and maximum. For example, `Range1-1000` would allow a number between 1 and 1000 to be selected; `Range-50-50` would allow a number between -50 and 50; and `Range-10--5` would allow a number between -10 and -5.

**Thick:** Specifies the length of the thumb (the bar moved by the user). After the word **Thick**, specify the thickness in pixels (e.g. `Thick30`). To go beyond a certain thickness on Windows XP or later, it is probably necessary to either specify the **Center** option or remove the theme from the control (which can be done by specifying `-Theme` in the control's options).

**TickInterval:** Provides tickmarks alongside the track at the specified interval. After the word **TickInterval**, specify the interval at which to display additional tickmarks (if the interval is omitted, it is assumed to be 1). For example, `TickInterval10` would display a tickmark once every 10 positions.

**ToolTip:** Creates a tooltip that reports the numeric position of the slider as the user is dragging it. To have the tooltip appear in a non-default position, specify

one of the following instead: `ToolTipLeft` or `ToolTipRight` (for vertical sliders); `ToolTipTop` or `ToolTipBottom` (for horizontal sliders).

**Vertical:** Makes the control slide up and down rather than left and right.

The above options can be changed after the control is created via [GuiCtrl.Opt](#).

## Progress

Description: A dual-color bar typically used to indicate how much progress has been made toward the completion of an operation. Example:

```
Gui.Add("Progress", "w300 h20 cBlue
vMyProgress")
```

Specify the starting position of the bar as the third parameter (if omitted, the bar starts off at 0 or the number in the allowable range that is closest to 0). To later change the position of the bar, follow these examples, all of which operate upon a progress bar whose `Name` is `MyProgress`:

```
Gui.Control["MyProgress"].Value += 20 ;
Increase the current position by 20.
Gui.Control["MyProgress"].Value := 50 ; Set
the current position to 50.
```

For horizontal Progress Bars, the thickness of the bar is equal to the control's height. For vertical Progress Bars it is equal to the control's width.

## Progress Options

**Cn:** Changes the bar's color. Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. Examples: `cRed`, `cFFFF33`, `cDefault`. If the **C** option is never used (or `cDefault` is specified), the system's default bar color will be used.

**BackgroundN:** Changes the bar's background color. Specify for **n** one of the 16 primary HTML [color names](#) or a 6-digit RGB color value. Examples:

`BackgroundGreen`, `BackgroundFFFF33`, `BackgroundDefault`. If the Background option is never used (or `BackgroundDefault` is specified), the background color will be that of the window or [tab control](#) behind it.

**Range:** Sets the range to be something other than 0 to 100. After the word Range, specify the minimum, a dash, and maximum. For example, `Range0-1000` would allow a numbers between 0 and 1000; `Range-50-50` would allow numbers between -50 and 50; and `Range-10--5` would allow numbers between -10 and -5.

**Smooth:** Displays a simple continuous bar. If this option is not used and the bar does not have any custom colors, the bar's appearance is defined by the current system theme. Otherwise, the bar appears as a length of segments.

**Vertical:** Makes the bar rise or fall vertically rather than move along horizontally.

The above options can be changed after the control is created via [GuiCtrl.Opt](#).

## GroupBox

Description: A rectangular border/frame, often used around other controls to indicate they are related. In this case, the last parameter is the title of the box, which if present is displayed at its upper-left edge. Example:

```
Gui.Add("GroupBox", "w400 h300", "Geographic
Criteria")
```

By default, a GroupBox's title may have only one line of text. This can be overridden by specifying `wrap` in Options.

To specify the number of rows inside the control (or its height and width), see [position and sizing of controls](#).

## Tab3

Description: A large control containing multiple pages, each of which contains other controls. From this point forward, these pages are referred to as "tabs".

There are three types of Tab control:

- **Tab3:** Fixes some issues which affect Tab2 and Tab. Controls are placed within an invisible "tab dialog" which moves and resizes with the tab control. The tab control is themed by default.
- **Tab2:** Fixes rare redrawing problems in the original "Tab" control but introduces [some other problems](#).
- **Tab:** Retained for backward compatibility because of [differences in behavior](#) between Tab2/Tab3 and Tab.

Example:

```
Gui.Add("Tab3", ,
"General|View|Appearance|Settings")
```

The last parameter above is a pipe-delimited list or an array of tab names. To have one of the tabs pre-selected when the window first appears, include two pipe characters after it (e.g. `Red|Green| |Blue`). Alternatively, include in *Options* the word **Choose** followed immediately by the number to pre-select. For example, `Choose5` would pre-select the fifth tab (as with other options, it can also be a variable such as `Choose%Var%`). After the control is created, use [Value](#), [Text](#) or [Choose](#) to change the selected tab, and [Add](#) or [Delete](#) to add or

remove tabs.

After creating a Tab control, subsequently added controls automatically belong to its first tab. This can be changed at any time by following these examples (in this case, *Tab* is the [GuiControl object](#) of the first tab control and *Tab2* of the second one):

```
Tab.UseTab() ; Future controls are not part of
any tab control.
Tab.UseTab(3) ; Future controls are owned by
the third tab of the current tab control.
Tab2.UseTab(3) ; Future controls are owned by
the third tab of the second tab control.
Tab.UseTab("Name") ; Future controls are owned
by the tab whose name starts with Name (not
case sensitive).
Tab.UseTab("Name", true) ; Same as above but
requires exact match (not case sensitive).
```

It is also possible to use any of the examples above to assign controls to a tab or tab-control that does not yet exist (except in the case of the *Name* method). But in that case, the relative positioning options described below are not supported.

**Positioning:** When each tab of a Tab control receives its first sub-control, that sub-control will have a special default position under the following conditions: 1) The X and Y coordinates are both omitted, in which case the first sub-control is positioned at the upper-left corner of the tab control's interior (with a standard [margin](#)), and sub-controls beyond the first are positioned beneath the previous control; 2) The [X+n](#) and/or [Y+n](#) positioning options are specified, in which case the sub-control is positioned relative to the upper-left corner of the tab control's

interior. For example, specifying `x+10 y+10` would position the control 10 pixels right and 10 pixels down from the upper left corner.

**Current tab:** `GuiCtrl.Value` returns the position number of the currently selected tab (the first tab is 1, the second is 2, etc.). To get its text instead, use `GuiCtrl.Text`. `Gui.Submit` stores `Text` by default, but stores `Value` instead if the word **AltSubmit** is in the control's *Options*.

**Detecting tab selection:** Whenever the user switches tabs, the `Change` event is raised.

**Keyboard navigation:** The user may press Control-PageDown/PageUp to navigate from page to page in a tab control; if the keyboard focus is on a control that does not belong to a Tab control, the window's first Tab control will be navigated. Control-Tab and Control-Shift-Tab may also be used except that they will not work if the currently focused control is a multi-line Edit control.

**Limits:** Each window may have no more than 255 tab controls. Each tab control may have no more than 256 tabs (pages). In addition, a tab control may not contain other tab controls.

## Tab3 vs. Tab2 vs. Tab

**Parent window:** The parent window of a control affects the positioning and visibility of the control and tab-key navigation order. If a sub-control is added to an existing Tab3 control, its parent window is the "tab dialog", which fills the tab control's display area. Most other controls, including sub-controls of Tab or Tab2

controls, have no parent other than the GUI window itself.

**Positioning:** For Tab and Tab2, sub-controls do not necessarily need to exist within their tab control's boundaries: they will still be hidden and shown whenever their tab is selected or de-selected. This behavior is especially appropriate for the "buttons" style described below.

For Tab3, sub-controls assigned to a tab *before* the tab control is created behave as though added to a Tab or Tab2 control. All other sub-controls are visible only within the display area of the tab control.

If a Tab3 control is moved, its sub-controls are moved with it. Tab and Tab2 controls do not have this behavior.

In the rare case that [WinMove](#) (or an equivalent DllCall) is used to move a control, the coordinates must be relative to the parent window of the control, which might not be the GUI (see [above](#)). By contrast, [GuiCtrl.Move](#) takes GUI coordinates and [ControlMove](#) takes window coordinates, regardless of the control's parent window.

**Autosizing:** If not specified by the script, the width and/or height of the Tab3 control are automatically calculated at one of the following times (whichever comes first after the control is created):

- The first time the Tab3 control ceases to be the current tab control. This can occur as a result of calling [GuiCtrl.UseTab](#) (with or without parameters) or creating another tab control.
- The first time [Gui.Show](#) is called for that particular Gui.

The calculated size accounts for sub-controls which exist when autosizing occurs, plus the default margins. The size is calculated only once, and will not be recalculated even if controls are added later. If the Tab3 control is empty, it receives the same default size as a Tab or Tab2 control.

Tab and Tab2 controls are not autosized; they receive an arbitrary default size.

**Tab-key navigation order:** The tab-key navigation order usually depends on the order in which the controls are created. When tab controls are used, the order also depends on the type of tab control:

- Tab and Tab2 allow their sub-controls to be mixed with other controls within the tab-key order.
- Tab2 puts its tab buttons after its sub-controls in the tab-key order.
- Tab3 groups its sub-controls within the tab-key order and puts them after its tab buttons.

**Notification messages (Tab3):** Common and Custom controls typically send notification messages to their parent window. Any WM\_COMMAND, WM\_NOTIFY, WM\_VSCROLL, WM\_HSCROLL or WM\_CTLCOLOR' messages received by a Tab3 control's tab dialog are forwarded to the GUI window and can be detected by using OnMessage. If the tab control is themed and the sub-control lacks the +BackgroundTrans option, WM\_CTLCOLORSTATIC is fully handled by the tab dialog and not forwarded. Other notification messages (such as custom messages) are not supported.

**Known issues with Tab2:**

- `BackgroundTrans` has no effect inside a `Tab2` control.
- `WebBrowser` controls do not redraw correctly.
- `AnimateWindow` and possibly other Win32 API calls can cause the tab's controls to disappear.

### Known issues with Tab:

- Activating a GUI window by clicking certain parts of its controls, such as scrollbars, might redraw improperly.
- `BackgroundTrans` has no effect if the `Tab` control contains a `ListView`.
- `WebBrowser` controls are invisible.

### Tab Options

**Choose:** See [above](#).

**-Background** (minus followed by the word background): Overrides the window's custom background color and uses the system's default `Tab` control color. Specify `+Theme -Background` to make the `Tab` control conform to the current desktop theme. However, most control types will look strange inside such a `Tab` control because their backgrounds will not match that of the tab control. This can be fixed for some control types (such as `Text`) by adding `BackgroundTrans` to their options.

**Buttons:** Creates a series of buttons at the top of the control rather than a series of tabs (in this case, there will be no border by default because the display area does not typically contain controls).

**Left/Right/Bottom:** Specify one of these words to have the tabs on the left, right, or bottom side instead of the top. See [TCS\\_VERTICAL](#) for limitations on Left and Right.

**-Wrap:** Prevents the tabs from taking up more than a single row (in which case if there are too many tabs to fit, arrow buttons are displayed to allow the user to slide more tabs into view).

To specify the number of rows of text inside the control (or its height and width), see [position and sizing of controls](#).

**Icons in Tabs:** An icon may be displayed next to each tab's name/text via [SendMessage](#). This is demonstrated in the forum topic [Icons in tabs](#).

## StatusBar

Description: A row of text and/or icons attached to the bottom of a window, which is typically used to report changing conditions. Example:

```
SB := Gui.Add("StatusBar",, "Bar's starting
text (omit to start off empty).")
SB.SetText("There are " . RowCount . " rows
selected.")
```

The simplest use of a status bar is to call `SB.SetText` whenever something changes that should be reported to the user. To report more than one piece of information, divide the bar into sections via `SB.SetParts`. To display icon(s) in the bar, call `SB.SetIcon`.

### SetText

Displays *NewText* in the specified part of the status bar, and returns 1 upon success and 0 upon failure.

```
Success := SB.SetText(NewText [, PartNumber := 1,
Style := 0])
```

### NewText

Up to two tab characters (``t`) may be present anywhere in *NewText*: anything to the right of the first tab is centered within the part, and anything to the right of the second tab is right-justified.

## PartNumber

If *PartNumber* is omitted, it defaults to 1. Otherwise, specify an integer between 1 and 256.

## Style

If *Style* is omitted, it defaults to 0, which uses a traditional border that makes that part of the bar look sunken. Otherwise, specify 1 to have no border or 2 to have border that makes that part of the bar look raised.

## SetParts

Divides the bar into multiple sections according to the specified widths (in pixels), and returns a non-zero value (the status bar's [HWND](#)).

```
Hwnd := SB.SetParts([Width1, Width2, ... Width255])
```

## Width1 ... Width255

If all parameters are omitted, the bar is restored to having only a single, long part. Otherwise, specify the width of each part except the last (the last will fill the remaining width of the bar). For example,

`SB.SetParts(50, 50)` would create three parts: the first two of width 50 and the last one of all the remaining width.

Note: Any parts "deleted" by `SB.SetParts()` will start off with no text the next time they are shown (furthermore, their icons are automatically

destroyed).

## SetIcon

Displays a small icon to the left of the text in the specified part, and returns the icon's handle.

```
HICON := SB.SetIcon(FileName [, IconNumber := 1, PartNumber := 1])
```

### FileName

*Filename* is the name of an icon (.ICO), cursor (.CUR), or animated cursor (.ANI) file (animated cursors will not actually be animated in the bar). Other sources of icons include the following types of files: EXE, DLL, CPL, SCR, and other types that contain icon resources. An [icon handle](#) can be used instead of a filename. For example,

```
SB.SetIcon("HICON:" handle).
```

### IconNumber

To use an icon group other than the first one in the file, specify its number for *IconNumber*. For example,

```
SB.SetIcon("Shell32.dll", 2)
```

would use the default icon from the second icon group. If *IconNumber* is negative, its absolute value is assumed to be the resource ID of an icon within an executable file.

## PartNumber

If *PartNumber* is omitted, it defaults to 1. Otherwise, specify an integer between 1 and 256.

Note: The HICON is a system resource that can be safely ignored by most scripts because it is destroyed automatically when the status bar's window is destroyed. Similarly, any old icon is destroyed when `SB.SetIcon()` replaces it with a new one. This can be avoided via:

```
Gui.Opt("+LastFound")
SendMessage(0x40F, part_number - 1, my_hIcon,
"msctls_statusbar321") ; 0x40F is SB_SETICON.
```

## SetProgress

Creates and controls a progress bar inside the status bar. This function is available at [www.autohotkey.com/forum/topic37754.html](http://www.autohotkey.com/forum/topic37754.html)

## Reacting to Mouse Clicks

Whenever the user clicks on the bar, the `Click`, `DoubleClick` or `ContextMenu` event is raised, and the *Info* or *Item* parameter contains the part number.

However, the part number might be a very large integer if the user clicks near the sizing grip at the right side of the bar.

## Font and Color

Although the font size, face, and style can be set via `Gui.SetFont` (just like normal controls), the text color cannot be changed. The status bar's background color may be changed by specifying in *Options* the word **Background** followed immediately by a color name (see [color chart](#)) or RGB value (the 0x prefix is optional). Examples: `BackgroundSilver`, `BackgroundFFDD99`, `BackgroundDefault`.

## Hiding the StatusBar

Upon creation, the bar can be hidden via `MyStatusBar := Gui.Add("StatusBar", "Hidden")`. To hide it sometime after creation, use `MyStatusBar.Visible := false`. To show it, use `MyStatusBar.Visible := true`. Note: Hiding the bar does not reduce the height of the window. If that is desired, one easy way is `Gui.Show("AutoSize")`.

## Styles (rarely used)

See the [StatusBar styles table](#).

## Known Limitations

1) Any control that overlaps the status bar might sometimes get drawn on top of it. One way to avoid this is to dynamically shrink such controls via [Size](#) event. 2) There is a limit of one status bar per window.

## Example

The bottom of the [TreeView page](#) demonstrates a multipart status bar.

## ActiveX

ActiveX components such as the MSIE browser control can be embedded into a GUI window by following this example:

```
Gui := GuiCreate()
WB := Gui.Add("ActiveX", "w980 h640",
"Shell.Explorer").Value ; The last parameter
is the name of the ActiveX component.
WB.Navigate("https://autohotkey.com/boards/")
; This is specific to the web browser control.
Gui.Show()
```

When the control is created, the ActiveX object can be retrieved via `GuiCtrl.Value`.

To handle events exposed by the object, use `ComObjConnect` as demonstrated below:

```
Gui := GuiCreate()
URL := Gui.Add("Edit", "w930 r1",
"https://autohotkey.com/boards/")
Gui.Add("Button", "x+6 yp w44 Default",
"Go").OnEvent("Click", "ButtonGo")
WB := Gui.Add("ActiveX", "xm w980 h640",
"Shell.Explorer").Value
Gui.Show()
; Continue on to load the initial page:
ButtonGo()

ButtonGo() {
 global
```

```
WB.Navigate(URL.Value)
}

class WB_events {
 NavigateComplete2(wb, NewURL) {
 global
 URL.Value := NewURL ; Update the URL edit
 control.
 }
}
```

`ComObjType` can be used to determine the type of the retrieved object.

## Custom

Other controls which are not directly supported by AutoHotkey can be also embedded into a GUI window. In order to do so, the Win32 class name must be specified through the `Class` option in `Gui.Add`. Examples:

```
Gui.Add("Custom", "ClassComboBoxEx32") ; Adds a ComboBoxEx control.
Gui.Add("Custom", "ClassScintilla") ; Adds a Scintilla control. Note that the SciLexer.dll library must be loaded before the control can be added.
```

AutoHotkey uses the standard Windows control text routines when text is to be retrieved/replaced in the control via `Gui.Add` or `GuiCtrl.Value`.

**Events:** Since the meaning of each notification code depends on the control which sent it, `OnEvent` is not supported for Custom controls. However, if the control sends notifications in the form of a `WM_NOTIFY` or `WM_COMMAND` message, the script can use `OnNotify` or `OnCommand` to detect them.

Here is an example that shows how to add and use an [IP address control](#):

```
Gui := GuiCreate()
; Add and set up the IP address control:
IP := Gui.Add("Custom", "ClassSysIPAddress32 r1 w150")
IPCtrlSetAddress(IP.Hwnd, A_IPAddress1)
IPText := Gui.Add("Text", "wp")
```

```

IPField := Gui.Add("Text", "wp y+m")
IP.OnCommand(0x300,
Func("IP_EditChange").bind(IPText)) ;
EN_CHANGE = 0x300
IP.OnNotify(-860,
Func("IP_FieldChange").bind(IPField)) ;
IPN_FIELDCHANGED = -860
fn := Func("OK_Click").bind(Gui, IP)
Gui.Add("Button", "Default",
"OK").OnEvent("Click", fn)
Gui.Show()

OK_Click(Gui, IP)
{
 Gui.Hide()
 MsgBox("You chose "
IPCtrlGetAddress(IP.Hwnd))
 ExitApp()
}

IP_EditChange(IPText, IP)
{
 IPText.Text := "New text: " IP.Text
}

IP_FieldChange(IPField, IP, nmipaddress)
{
 ; Extract info from the NMIPADDRESS
structure.
 iField := NumGet(nmipaddress + 3*A_PtrSize
+ 0, "int")
 iValue := NumGet(nmipaddress + 3*A_PtrSize
+ 4, "int")
 if iValue >= 0
 IPField.Text := "Field #" iField "
modified: " iValue
 else

```

```

 IPField.Text := "Field #" iField " left
empty"
 }

IPCtrlSetAddress(hControl, ipaddress)
{
 static WM_USER := 0x400
 static IPM_SETADDRESS := WM_USER + 101

 ; Pack the IP address into a 32-bit word
for use with SendMessage.
 ipaddrword := 0
 LoopParse(ipaddress, ".")
 ipaddrword := (ipaddrword * 256) +
A_LoopField
 SendMessage(IPM_SETADDRESS, 0, ipaddrword,,
"ahk_id " hControl)
}

IPCtrlGetAddress(hControl)
{
 static WM_USER := 0x400
 static IPM_GETADDRESS := WM_USER + 102

 VarSetCapacity(addrword, 4)
 SendMessage(IPM_GETADDRESS, 0, &addrword,,
"ahk_id " hControl)
 return NumGet(addrword, 3, "UChar") "."
NumGet(addrword, 2, "UChar") "."
NumGet(addrword, 1, "UChar") "."
NumGet(addrword, 0, "UChar")
}

```

## Related Pages

[ListView](#), [TreeView](#), [GuiCreate](#), [Gui object](#), [GuiControl object](#), [Menu](#)

# GUI Object

Provides an interface for creating and managing windows, and creating controls. Such windows can be used as data entry forms or custom user interfaces. `GuiCreate` and `GuiFromHwnd` returns an object of this type.

## Properties:

- **BackColor:** Retrieves or sets the background color of the window.
- **ClientPos:** Retrieves the position and size of the window's client area.
- **Control:** Retrieves the `GuiControl` object associated with the specified name, `ClassNN` or `HWND`.
- **FocusedCtrl:** Retrieves the `GuiControl` object of the GUI's focused control.
- **Hwnd:** Retrieves the window handle (`HWND`) of the GUI window.
- **MarginX:** Retrieves or sets the size of horizontal margins between sides and subsequently created controls.
- **MarginY:** Retrieves or sets the size of vertical margins between sides and subsequently created controls.
- **Menu:** Adds or removes a menu bar.
- **Name:** Retrieves or sets a custom name for the GUI window.
- **Pos:** Retrieves the position and size of the window.
- **Title:** Retrieves or sets the GUI's title.

## Methods:

- **Add:** Creates a control such as text, button, or checkbox.
- **Destroy:** Deletes the window.

- **Flash:** Blinks the window and its taskbar button.
- **Hide / Cancel:** Hides the window.
- **Maximize:** Unhides and maximizes the window.
- **Minimize:** Unhides and minimizes the window.
- **\_NewEnum:** Iterates through the GUI's controls.
- **OnEvent:** Registers a function or method to be called when the given event is raised.
- **Opt:** Sets various options and styles for the appearance and behavior of the window.
- **Restore:** Unhides and restores the window, if it was minimized or maximized beforehand.
- **SetFont:** Sets the typeface, size, style, and text color for subsequently created controls.
- **Show:** Displays the window. It can also minimize, maximize, or move the window.
- **Submit:** Saves the user's input and optionally hides the window.

# Add

Adds a control to the GUI window, and returns a [GuiControl](#) object.

```
Gui.Add(ControlType [, Options, Text])
Gui.AddControlType([Options, Text])
```

## ControlType

This is one of the following: [Text](#), [Edit](#), [UpDown](#), [Picture](#), [Button](#), [Checkbox](#), [Radio](#), [DropDownList](#), [ComboBox](#), [ListBox](#), [ListView](#), [TreeView](#), [Link](#), [Hotkey](#), [DateTime](#), [MonthCal](#), [Slider](#), [Progress](#), [GroupBox](#), [Tab](#), [StatusBar](#), [ActiveX](#), [Custom](#)

For example:

```
Gui := GuiCreate()
Gui.Add("Text",, "Please enter your
name:")
Gui.AddEdit("vName")
Gui.Show
```

## Options

### Positioning and Sizing of Controls

If some dimensions and/or coordinates are omitted from *Options*, the control will be positioned relative to the previous control and/or sized automatically according to its nature and contents.

The following options are supported:

**R:** Rows of text (can contain a floating point number such as R2.5). **R** is often preferable to specifying **H** (Height). If both the **R** and **H** options are present, **R** will take precedence. For a GroupBox, this setting is the number of controls for which to reserve space inside the box. For DropDownLists, ComboBoxes, and ListBoxes, it is the number of items visible at one time inside the list portion of the control (but on Windows XP or later, it is often desirable to omit both the **R** and **H** options for DropDownList and ComboBox, which makes the popup list automatically take advantage of the available height of the user's desktop). For other control types, **R** is the number of rows of text that can visibly fit inside the control.

**W:** Width, in pixels. If omitted, the width is calculated automatically for some control types based on their contents. The other controls types have the following default widths:

Tab controls: 30 times the current font size, plus 3 times the *X-margin*.

Vertical Progress Bars: Two times the current font size.

Horizontal Progress Bars, horizontal Sliders, DropDownLists, ComboBoxes, ListBoxes, GroupBoxes, Edits, and Hotkeys: 15 times the current font size (except GroupBoxes, which multiply by 18 to provide room inside for margins).

**H:** Height, in pixels. If both the **H** and **R** options are absent, DropDownLists, ComboBoxes, ListBoxes, and empty multi-line Edit controls default to 3 rows; GroupBoxes default to 2 rows; vertical

Sliders and Progress Bars default to 5 rows; horizontal Sliders default to 30 pixels (except if a thickness has been specified); horizontal Progress Bars default to 2 times the current font size; Hotkey controls default to 1 row; and Tab controls default to 10 rows. For the other control types, the height is calculated automatically based on their contents. Note that for DropDownLists and ComboBoxes, **H** is the combined height of the control's always-visible portion and its list portion (but even if the height is set too low, at least one item will always be visible in the list). Also, for all types of controls, specifying the number of rows via the **R** option is usually preferable to using **H** because it prevents a control from showing partial/incomplete rows of text.

**wp+n**, **hp+n**, **wp-n**, **hp-n** (where **n** is any number) can be used to set the width and/or height of a control equal to the previously added control's width or height, with an optional plus or minus adjustment. For example, **wp** would set a control's width to that of the previous control, and **wp-50** would set it equal to 50 less than that of the previous control.

**X**: X-position. For example, specifying **x0 y0** would position the control in the upper left corner of the window's client area, which is the area beneath the title bar and menu bar (if any). If **X** is omitted but not **Y**, the control will be positioned to the right of all previously added controls, which can be thought of as starting a new "column".

**Y**: Y-position. If **Y** is omitted but not **X**, the control will be positioned beneath all previously added controls, which can be thought of as starting a new "row".

Omitting either **X**, **Y** or both is useful to make a GUI layout automatically adjust to any future changes you might make to the size of controls or font. By contrast, specifying an absolute position for every control might require you to manually shift the position of all controls that lie beneath and/or to the right of a control that is being enlarged or reduced.

If both **X** and **Y** are omitted, the control will be positioned beneath the previous control using a standard padding distance.

For **X** and **Y**, an optional plus sign can be included to position a control relative to the right or bottom edge (respectively) of the control that was previously added. For example, specifying `Y+10` would position the control 10 pixels beneath the bottom of the previous control rather than using the standard padding distance. Similarly, specifying `X+10` would position the control 10 pixels to the right of the previous control's right edge. Since negative numbers such as `X-10` are reserved for absolute positioning, to use a negative offset, include a plus sign in front of it. For example: `X+-10`.

For **X+** and **Y+**, the letter **M** can be used as a substitute for the window's current `margin`. For example, `x+m` uses the right edge of the previous control plus the standard padding distance. `xp y+m` positions

a control below the previous control, whereas specifying an X coordinate on its own would normally imply **yp** by default.

**xp+n**, **yp+n**, **xp-n**, **yp-n** (where **n** is any number) can be used to position controls relative to the previous control's upper left corner, which is often useful for enclosing controls in a [GroupBox](#).

**xm** and **ym** can be used to position a control at the leftmost and topmost [margins](#) of the window, respectively (these two may also be followed by a plus/minus sign and a number). By specifying **ym** without any x-position at all, the control will be positioned at the top margin but to the right of all previously added controls, which can be thought of as starting a new "column". The converse is also true.

**xs** and **ys**: these are similar to **xm** and **ym** except that they refer to coordinates that were saved by having previously added a control with the word [Section](#) in its options (the first control of the window always starts a new section, even if that word isn't specified in its options). By specifying **ys** without any x-position at all, the control will be positioned at the previously saved y-position, but to the right of all controls that have been added since the most recent use of the word [Section](#); this can be thought of as starting a new column within the section. For example:

```
Gui := GuiCreate()
Gui.Add("Edit", "w600") ; Add a fairly
wide edit control at the top of the
window.
```

```
Gui.Add("Text", 'section', "First Name:")
; Save this control's position and start
a new section.
Gui.Add("Text",, "Last Name:")
Gui.Add("Edit", 'ys') ; Start a new
column within this section.
Gui.Add("Edit")
Gui.Show()
```

The converse of the above (specifying `xs` but omitting the `y`-position) is also true.

`xs` and `ys` may optionally be followed by a plus/minus sign and a number. Also, it is possible to specify both the word `Section` and `xs/ys` in a control's options; this uses the previous section for itself but establishes a new section for subsequent controls.

### Storing and Responding to User Input

**V:** Sets the control's `Name`. Specify the name immediately after the letter `V`, which is not included in the name. For example, specifying `vMyEdit` would name the control "MyEdit".

**Events:** Event handlers (such as a function which is called automatically when the user clicks or changes a control) cannot be set within the control's *Options*. Instead, `OnEvent` can be used to register a callback function or method for each event of interest.

### Controls: Common Styles and Other Options

Note: In the absence of a preceding sign, a plus sign is assumed; for

example, `wrap` is the same as `+wrap`. By contrast, `-wrap` would remove the word-wrapping property.

**AltSubmit:** Uses alternate submit method. For `DropDownList`, `ComboBox`, `ListBox` and `Tab`, this causes `Gui.Submit` to store the position of the selected item rather than its text. If no item is selected, a `ComboBox` will still store the text of its edit field.

**C:** Color of text (has no effect on `buttons`). Specify the letter `C` followed immediately by a color name (see `color chart`) or RGB value (the `0x` prefix is optional). Examples: `cRed`, `cFF2211`, `c0xFF2211`, `cDefault`.

**Disabled:** Makes an input-capable control appear in a disabled state, which prevents the user from focusing or modifying its contents. Use `GuiCtrl.Enabled` to enable it later. Note: To make an Edit control read-only, specify the string `ReadOnly` instead. Also, the word `Disabled` may optionally be followed immediately by a 0 or 1 to indicate the starting state (0 for enabled and 1 for disabled). In other words, `Disabled` and `Disabled%VarContainingOne%` are the same.

**Hidden:** The control is initially invisible. Use `GuiCtrl.Visible` to show it later. The word `Hidden` may optionally be followed immediately by a 0 or 1 to indicate the starting state (0 for visible and 1 for hidden). In other words, `Hidden` and `Hidden%VarContainingOne%` are the same.

**Left:** Left-justifies the control's text within its available width. This option affects the following controls: Text, Edit, Button, Checkbox, Radio, UpDown, Slider, Tab, Tab2, GroupBox, DateTime.

**Right:** Right-justifies the control's text within its available width. For checkboxes and radio buttons, this also puts the box itself on the right side of the control rather than the left. This option affects the following controls: Text, Edit, Button, Checkbox, Radio, UpDown, Slider, Tab, Tab2, GroupBox, DateTime, Link.

**Center:** Centers the control's text within its available width. This option affects the following controls: Text, Edit, Button, Checkbox, Radio, Slider, GroupBox.

**Section:** Starts a new section and saves this control's position for later use with the `xs` and `ys` positioning options described [above](#).

**Tabstop:** Use `-Tabstop` (i.e. minus Tabstop) to have an input-capable control skipped over when the user presses the TAB key to navigate.

**Wrap:** Enables word-wrapping of the control's contents within its available width. Since nearly all control types start off with word-wrapping enabled, use `-wrap` to disable word-wrapping.

**VScroll:** Provides a vertical scroll bar if appropriate for this type of control.

**HScroll:** Provides a horizontal scroll bar if appropriate for this type of control. The rest of this paragraph applies to [ListBox](#) only. The horizontal scrolling width defaults to 3 times the width of the [ListBox](#). To specify a different scrolling width, include a number immediately after the word `HScroll`. For example, `HScroll1500` would allow 500 pixels of scrolling inside the [ListBox](#). However, if the specified scrolling width is smaller than the width of the [ListBox](#), no scroll bar will be shown (though the mere presence of *HScroll* makes it possible for the horizontal scroll bar to be added later via `MyScrollBar.Opt("+HScroll1500")`, which is otherwise impossible).

### Controls: Uncommon Styles and Options

**BackgroundTrans:** Uses a transparent background, which allows any control that lies behind a [Text](#), [Picture](#), or [GroupBox](#) control to show through. For example, a transparent [Text](#) control displayed on top of a [Picture](#) control would make the text appear to be part of the picture. Use `GuiCtrl.Opt("+Background")` to remove this option later. See [Picture control's AltSubmit section](#) for more information about transparent images. Known limitation: `BackgroundTrans` might not work properly for controls inside a [Tab control](#) that contains a [ListView](#). If a control type does not support this option, an error is thrown.

**BackgroundColor:** Changes the background color of the control. Replace *Color* with a color name (see [color chart](#)) or RGB value (the

0x prefix is optional). Examples: `BackgroundSilver`, `BackgroundFFDD99`. If this option is not present, a `Text`, `Picture`, `GroupBox`, `CheckBox`, `Radio`, `Slider`, `Tab` or `Link` control initially defaults to the background color set by `Gui.BackColor` (or if none or other control type, the system's default background color). Specifying `BackgroundDefault` or `-Background` applies the system's default background color. For example, a control can be restored to the default color via `LV.Opt("+BackgroundDefault")`. Using `+Background` without specifying a color reverts `-Background`. If a control type does not support this option, an error is thrown.

**Border:** Provides a thin-line border around the control. Most controls do not need this because they already have a type-specific border. When adding a border to an *existing* control, it might be necessary to increase the control's width and height by 1 pixel.

**Theme:** This option can be used to override the window's current theme setting for the newly created control. It has no effect when used on an existing control; however, this may change in a future version. See GUI's `+/-Theme` option for details.

**(Unnamed Style):** Specify a plus or minus sign followed immediately by a decimal or hexadecimal `style number`. If the sign is omitted, a plus sign is assumed.

**(Unnamed ExStyle):** Specify a plus or minus sign followed immediately by the letter E and a decimal or hexadecimal extended

style number. If the sign is omitted, a plus sign is assumed. For example, `E0x200` would add the `WS_EX_CLIENTEDGE` style, which provides a border with a sunken edge that might be appropriate for pictures and other controls. Although the other extended styles are not documented here (since they are rarely used), they can be discovered by searching for `WS_EX_CLIENTEDGE` at [www.microsoft.com](http://www.microsoft.com).

### Text

Depending on the specified control type, a string, number or an array.

# Show

By default, this makes the window visible, unminimizes it (if necessary) and activates it.

```
Gui.Show([Options])
```

## Options

Omit the X, Y, W, and H options below to have the window retain its previous size and position. If there is no previous position, the window will be auto-centered in one or both dimensions if the X and/or Y options mentioned below are absent. If there is no previous size, the window will be auto-sized according to the size and positions of the controls it contains.

Zero or more of the following strings may be present in *Options* (specify each number as decimal, not hexadecimal):

**Wn**: Specify for **n** the width (in pixels) of the window's client area (the client area excludes the window's borders, title bar, and [menu bar](#)).

**Hn**: Specify for **n** the height of the window's client area, in pixels.

**Xn**: Specify for **n** the window's X-position on the screen, in pixels. Position 0 is the leftmost column of pixels visible on the screen.

**Yn**: Specify for **n** the window's Y-position on the screen, in pixels. Position 0 is the topmost row of pixels visible on the screen.

**Center:** Centers the window horizontally and vertically on the screen.

**xCenter:** Centers the window horizontally on the screen. For example:

```
Gui.Show("xCenter y0").
```

**yCenter:** Centers the window vertically on the screen.

**AutoSize:** Resizes the window to accommodate only its currently visible controls. This is useful to resize the window after new controls are added, or existing controls are resized, hidden, or unhidden. For example:

```
Gui.Show("AutoSize Center")
```

*One of the following may also be present:*

**Minimize:** Minimizes the window and activates the one beneath it.

**Maximize:** Maximizes and activates the window.

**Restore:** Unminimizes or unmaximizes the window, if necessary. The window is also shown and activated, if necessary.

**NoActivate:** Unminimizes or unmaximizes the window, if necessary. The window is also shown without activating it.

**NA:** Shows the window without activating it. If the window is minimized, it will stay that way but will probably rise higher in the z-order (which is the order seen in the alt-tab selector). If the window was previously hidden, this will probably cause it to appear on top of

the active window even though the active window is not deactivated.

**Hide:** Hides the window and activates the one beneath it. This is identical in function to [Gui.Hide](#) except that it allows a hidden window to be moved or resized without showing it. For example:

```
Gui.Show("Hide x55 y66 w300 h200").
```

## Submit

Saves the contents of controls into an [associative array](#) (object) and returns it. By default, also hides the window.

```
NamedCtrlContents := Gui.Submit([Hide := true])
```

### Hide

If this parameter is false, the window will not be hidden.

The returned array contains one element per control, usually like `array[GuiCtrl.Name] := GuiCtrl.Value`, with the exceptions noted below. Only input-capable controls which support [GuiCtrl.Value](#) and have been given a name are included.

For [DropDownList](#), [ComboBox](#), [ListBox](#) and [Tab](#), the text of the selected item/tab is stored instead of its position number if the control **lacks** the [AltSubmit](#) option, or if the [ComboBox](#)'s text does not match a list item. Otherwise, [Value](#) (the item's position number) is stored.

If only one [Radio](#) button in a radio group has a name, [Submit](#) stores the number of the currently selected button instead of the control's [Value](#). 1 is the first radio button (according to original creation order), 2 is the second, and so on. If there is no button selected, 0 is stored.

Excluded because they are not input-capable: [Text](#), [Pic](#), [GroupBox](#), [Button](#), [Progress](#), [Link](#), [StatusBar](#).

Also excluded: [ListView](#), [TreeView](#), [ActiveX](#), [Custom](#).

## Hide / Cancel

Hides the window.

```
Gui.Hide()
Gui.Cancel()
```

## Destroy

Removes the window and all its controls, freeing the corresponding memory and system resources.

```
Gui.Destroy()
```

Note: If the script later recreates the window, all of the window's properties such as color and font will start off at their defaults (as though the window never existed). If `Gui.Destroy()` is not used, all GUI windows are automatically destroyed when the script exits.

## SetFont

Sets the font typeface, size, style, and/or color for controls added to the window from this point onward.

Note: Omit both parameters to restore the font to the system's default GUI typeface, size, and color. Otherwise, any font attributes which are not specified will be copied from the previous font.

```
Gui.SetFont([Options, FontName])
```

### Options

Zero or more options. Each option is either a single letter immediately followed by a value, or a single word. To specify more than one option, include a space between each. For example: `cBlue s12 bold`.

The following words are supported: **bold**, *italic*, ~~strike~~, underline, and *norm*. *Norm* returns the font to normal weight/boldness and turns off italic, strike, and underline (but it retains the existing color and size). It is possible to use *norm* to turn off all attributes and then selectively turn on others. For example, specifying `norm italic` would set the font to normal then to italic.

**C**: Color name (see [color chart](#)) or RGB value -- or specify the word *Default* to return to the system's default color (black on most systems). Example values: `cRed`, `cFFFFFFAA`, `cDefault`. Note: [Buttons](#) do not obey custom colors. Also, an individual control can be created with a

font color other than the current one by including the C option. For example: `Gui.Add("Text", "cRed", "My Text")`.

**S:** Size (in points). For example: `s12` (specify decimal, not hexadecimal)

**W:** Weight (boldness), which is a number between 1 and 1000 (400 is normal and 700 is bold). For example: `w600` (specify decimal, not hexadecimal)

**Q:** Text rendering quality. For example: `q3`. Q should be followed by a number from the following table:

|                            |                                                                                                                   |
|----------------------------|-------------------------------------------------------------------------------------------------------------------|
| 0 = DEFAULT_QUALITY        | Appearance of the font does not matter.                                                                           |
| 1 = DRAFT_QUALITY          | Appearance of the font is less important than when the PROOF_QUALITY value is used.                               |
| 2 = PROOF_QUALITY          | Character quality of the font is more important than exact matching of the logical-font attributes.               |
| 3 = NONANTIALIASED_QUALITY | Font is never antialiased, that is, font smoothing is not done.                                                   |
| 4 = ANTIALIASED_QUALITY    | Font is antialiased, or smoothed, if the font supports it and the size of the font is not too small or too large. |
|                            | Windows XP and later: If set, text is rendered (when                                                              |

5 = CLEARTYPE\_QUALITY

possible) using ClearType antialiasing method.

For more details of what these values mean, see [MSDN: CreateFont](#).

Since the highest quality setting is usually the default, this feature is more typically used to disable anti-aliasing in specific cases where doing so makes the text clearer.

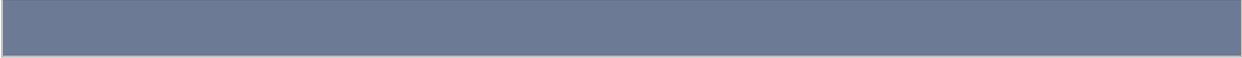
## FontName

*FontName* may be the name of any font, such as one from the [font table](#). If *FontName* is omitted or does not exist on the system, the previous font's typeface will be used (or if none, the system's default GUI typeface). This behavior is useful to make a GUI window have a similar font on multiple systems, even if some of those systems lack the preferred font. For example, by using the following commands in order, Verdana will be given preference over Arial, which in turn is given preference over MS sans serif:

```
Gui.SetFont(, "MS sans serif")
Gui.SetFont(, "Arial")
Gui.SetFont(, "Verdana") ; Preferred
font.
```

On a related note, the operating system offers standard dialog boxes that prompt the user to pick a font, color, or icon. These dialogs can be displayed via [DllCall\(\)](#) as demonstrated at

[www.autohotkey.com/forum/topic17230.html](http://www.autohotkey.com/forum/topic17230.html).



# BackColor

Retrieves or sets the background color of the window.

```
RetrievedColor := Gui.BackColor
```

```
Gui.BackColor := NewColor
```

*RetrievedColor* is a 6-digit RGB value of the current color previously set by this property.

*NewColor* is one of the 16 primary [HTML color names](#), a hexadecimal RGB color value (the 0x prefix is optional), a pure numeric RGB color value, or the word Default for its default color. Example values: `"Silver"`, `"FFFFFFAA"`, `0xFFFFAA`, `"Default"`.

By default, the window's background color is the system's color for the face of buttons.

The color of the [menu bar](#) and its submenus can be changed as in this example: `Menu, MyMenuBar, Color, White`.

To make the background transparent, use [WinSetTransColor](#). However, if you do this without first having assigned a custom window via `Gui.BackColor()`, buttons will also become transparent. To prevent this, first assign a custom color and then make that color transparent. For example:

```
Gui.BackColor("EEAA99")
WinSetTransColor("EEAA99")
```

To additionally remove the border and title bar from a window with a transparent background, use the following **after** the window has been made transparent: `Gui.Opt("-Caption")`

To illustrate the above, there is an example of an on-screen display (OSD) near the bottom of this page.

## MarginX

Retrieves or sets the size of horizontal margins between sides and subsequently created controls.

```
RetrievedValue := Gui.MarginX
```

```
Gui.MarginX := NewValue
```

*RetrievedValue* is the number of pixels of the current horizontal margin.

*NewValue* is the number of pixels of space to leave at the left and right side of the window when auto-positioning any control that lacks an explicit **X coordinate**. Also, the margin is used to determine the horizontal distance that separates auto-positioned controls from each other. Finally, the margin is taken into account by the first use of **Gui.Show** to calculate the window's size (when no explicit size is specified).

By default, this margin is proportional to the size of the currently selected **font** (1.25 times font-height for left & right).

## MarginY

Retrieves or sets the size of vertical margins between sides and subsequently created controls.

```
RetrievedValue := Gui.MarginY
```

```
Gui.MarginY := NewValue
```

*RetrievedValue* is the number of pixels of the current vertical margin.

*NewValue* is the number of pixels of space to leave at the top and bottom side of the window when auto-positioning any control that lacks an explicit [Y coordinate](#). Also, the margin is used to determine the vertical distance that separates auto-positioned controls from each other. Finally, the margin is taken into account by the first use of [Gui.Show](#) to calculate the window's size (when no explicit size is specified).

By default, this margin is proportional to the size of the currently selected [font](#) (0.75 times font-height for top & bottom).

# Opt

Sets one or more options for the GUI window.

```
Gui.Opt(Options)
Gui.Options(Options)
```

## Options

This parameter can contain any of the [options](#) supported by [GuiCreate](#).

For performance reasons, it is better to set all options before creating the window (that is, before any use of other methods such as [Gui.Add](#)).

## FocusedCtrl

Retrieves the [GuiControl](#) object of the GUI's focused control.

```
GuiCtrlObj := Gui.FocusedCtrl
```

Note: To be effective, the window generally must not be minimized or hidden.

# Menu

Attaches a menu bar to the window.

```
Gui.Menu := MenuName
```

*MenuName* is the name of an ordinary menu created by the [Menu](#) command.

For example:

```
Menu("FileMenu", "Add", "&Open tCtrl+O",
"MenuFileOpen") ; See remarks below about
Ctrl+O.
Menu("FileMenu", "Add", "E&xit",
"MenuHandler")
Menu("FileMenu", "Add", "&About",
"MenuHandler")
Menu("MyMenuBar", "Add", "&File", ":FileMenu")
; Attach the two sub-menus that were created
above.
Menu("MyMenuBar", "Add", "&Help", ":HelpMenu")
Gui.Menu := "MyMenuBar"
```

In the first line above, notice that `&Open` is followed by `Ctrl+O` (with a tab character in between). This indicates a keyboard shortcut that the user may press instead of selecting the menu item. If the shortcut uses only the standard modifier key names Ctrl, Alt and Shift, it is automatically registered as a *keyboard accelerator* for the GUI. Single-character accelerators with no modifiers are case-sensitive and can be triggered by unusual means such as IME or Alt+N<sub>NNN</sub>.

If a particular key combination does not work automatically, the use of a [context-sensitive hotkey](#) may be required. However, such hotkeys typically cannot be triggered by [Send](#) and are more likely to interfere with other scripts than a standard keyboard accelerator.

To remove a window's current menu bar, use `Gui.Menu := ""` (that is, assign an empty string).

Once a menu has been used as a menu bar, it should not be used as a popup menu or a submenu. This is because menu bars internally require a different format (however, this restriction applies only to the menu bar itself, not its submenus). If you need to work around this, create one menu to use as the menu bar and another identical menu to use for everything else.

The use of certain destructive [menu sub-commands](#) such as `Delete` and `DeleteAll` against a menu that is currently being used as a menu bar (and in some cases, its submenus) is not supported and will cause an error dialog to be displayed (unless [UseErrorLevel](#) is in effect). Use the following steps to make such changes: 1) detach the menu bar via `Gui.Menu := ""`; 2) make the changes; 3) reattach the menu bar via `Gui.Menu := "MyMenuBar"`.

## Minimize

Unhides the window (if necessary) and minimizes it.

```
Gui.Minimize()
```

# Maximize

Unhides the window (if necessary) and maximizes it.

```
Gui.Maximize()
```

## Restore

Unhides the window (if necessary) and restores it, if it was minimized or maximized beforehand.

```
Gui.Restore()
```

## Pos

Retrieves the position and size of the window.

```
PosSizeObj := Gui.Pos
```

*PosSizeObj* is an **object** with the keys **X** (x coordinate), **Y** (y coordinate), **W** (width) and **H** (height). The coordinates are the upper left corner of the window. Width is the horizontal distance between the left and right side of the window, and height the vertical distance between the top and bottom side (in pixels).

## ClientPos

Retrieves the position and size of the window's client area.

```
PosSizeObj := Gui.ClientPos
```

*PosSizeObj* is an [object](#) with the keys **X** (x coordinate), **Y** (y coordinate), **W** (width) and **H** (height). The coordinates are the upper left corner of the window's client area, which is the area not including title bar, menu bar, and borders. Width is the horizontal distance between the left and right side of the client area, and height the vertical distance between the top and bottom side (in pixels).

# Flash

Blinks the window's button in the taskbar.

```
Gui.Flash(Blink := true)
```

## Blink

If this parameter is omitted or true, the window's button in the taskbar will blink. This is done by inverting the color of the window's title bar and/or taskbar button (if it has one). Specify false to restore the original colors of the title bar and taskbar button (but the actual behavior might vary depending on OS version).

In the below example, the window will blink three times because each pair of flashes inverts then restores its appearance:

```
Loop(6)
{
 Gui.Flash()
 Sleep(500) ; It's quite sensitive to this
 value; altering it may change the behavior in
 unexpected ways.
}
```

## Hwnd

Retrieves the window handle (HWND) of the GUI window.

```
CurrentHwnd := Gui.Hwnd
```

A GUI's HWND is often used with [PostMessage](#), [SendMessage](#), and [DllCall](#). It can also be used directly as an [ahk\\_id WinTitle](#).

## Title

Retrieves or sets the GUI's title.

```
RetrievedTitle := Gui.Title
```

```
Gui.Title := NewTitle
```

## Name

Retrieves or sets a custom name for the GUI window.

```
RetrievedName := Gui.Name
```

```
Gui.Name := NewName
```

## Control

Retrieves the `GuiControl` object associated with the specified name, ClassNN or HWND.

```
GuiCtrlObj := Gui.Control[Name]
```

## **\_NewEnum**

Iterates through the GUI's controls.

```
Gui._NewEnum()
```

Note: The first output variable is the HWND, and the second is the [GuiControl](#) object.

# GuiControl Object

Provides an interface for modifying GUI controls and retrieving information about them. [Gui.Add](#), [Gui.Control](#), [GuiCtrlFromHwnd](#) and [Gui.\\_NewEnum](#) returns an object of this type.

## Properties:

- **ClassNN**: Retrieves the ClassNN of the control.
- **Enabled**: Retrieves the current interaction state of the control, or enables or disables (grays out) it.
- **Focused**: Retrieves the current focus state of the control.
- **Gui**: Retrieves the control's GUI parent.
- **Hwnd**: Retrieves the HWND of the control.
- **Name**: Retrieves or sets the explicit name of the control.
- **Pos**: Retrieves the position and size of the control.
- **Text**: Retrieves or sets the text/caption of the control.
- **Type**: Retrieves the type of the control.
- **Value**: Retrieves or sets new contents into a value-capable control.
- **Visible**: Retrieves the current visibility state of the control, or shows or hides it.

## Methods:

- **Add**: Appends the specified entries at the current list of a ListBox, DropDownList, ComboBox, or Tab control.
- **Choose**: Sets the selection in a ListBox, DropDownList, ComboBox, or Tab

control to be the specified value.

- **Delete:** Deletes the specified entry or all entries of a `ListBox`, `DropDownList`, `ComboBox`, or `Tab` control.
- **Focus:** Sets keyboard focus to the control.
- **Move:** Moves and/or resizes the control.
- **OnEvent:** Registers a function or method to be called when the given event is raised.
- **Opt:** Sets various options and styles for the appearance and behavior of the control.
- **SetFont:** Sets the font typeface, size, style, and/or color for the control.
- **SetFormat:** Sets the display format of a `DateTime` control.
- **UseTab:** Causes subsequently added controls to belong to the specified tab of a `Tab` control.

## Type

Retrieves the type of the control.

```
Type := GuiCtrl.Type
```

For a full list of control types, see [GUI control types](#).

## Hwnd

Retrieves the window handle (HWND) of the control.

```
Hwnd := GuiCtrl.Hwnd
```

A control's HWND is often used with [PostMessage](#), [SendMessage](#), and [DllCall](#).

## Name

Retrieves or sets the name of the control.

```
RetrievedName := GuiCtrl.Name
```

```
GuiCtrl.Name := NewName
```

The control's name can be used with [Gui.Control](#) to retrieve the [GuiControl](#) object. For most input-capable controls, the name is also used by [Gui.Submit](#).

## ClassNN

Retrieves the ClassNN of the control.

```
ClassNN := GuiCtrl.ClassNN
```

## Gui

Retrieves the [Gui object](#) of the control's parent GUI window.

```
GuiObj := GuiCtrl.Gui
```

# Opt

Adds or removes various options and styles of the control.

```
GuiCtrl.Opt(Options)
GuiCtrl.Options(Options)
```

## Options

Either [control-specific](#) or [general](#) options and styles.

In the following example, the control is [disabled](#) and its [background](#) is restored to the system default:

```
MyEdit.Opt("+Disabled -Background")
```

In the next example, the OK button is made the new default button:

```
OKButton.Opt("+Default")
```

Although [styles](#) and extended styles are also recognized, some of them cannot be applied or removed after a control has been created. ErrorLevel is set to 0 if at least one of the specified changes was successfully applied. Otherwise, it is set to 1 to indicate that none of the changes could be applied. Even if a change is successfully applied, the control might choose to ignore it.

# Move

Moves and/or resizes the control, optionally with drawing it again.

```
GuiCtrl.Move(Pos |, Draw := false|)
```

## Pos

One or more of the following option letters (specify each number as decimal, not hexadecimal):

**Xn**: Specify for **n** the new x-coordinate relative to the GUI window's client area, which is the area not including title bar, menu bar, and borders.

**Yn**: Specify for **n** the new y-coordinate relative to the GUI window's client area (see above).

**Wn**: Specify for **n** the new width of the control.

**Hn**: Specify for **n** the new height of the control.

## Draw

If this parameter is true, the region of the GUI window occupied by the control will be repainted. Although this may cause an unwanted flickering effect when called repeatedly and rapidly, it solves painting artifacts for certain control types such as [GroupBoxes](#).

## Examples

```
MyEdit.Move("x10 y20 w200 h100", true)
MyEdit.Move("x%VarX+10% y%VarY+5% w%VarW*2%
h%VarH*1.5%")
```

## Focus

Sets keyboard focus to the control.

```
GuiCtrl.Focus()
```

Note: To be effective, the window generally must not be minimized or hidden.

## Choose

Sets the selection in a `ListBox`, `DropDownList`, `ComboBox`, or `Tab` control to be the specified value.

```
GuiCtrl.Choose(Value)
```

### Value

This parameter should be 1 for the first entry, 2 for the second, etc.

If *Value* is a string (even a numeric string), the entry whose leading part matches *Value* will be selected. The search is not case sensitive.

For example, if the control contains the item "UNIX Text", specifying the word `unix` (lowercase) would be enough to select it. For a [multi-select `ListBox`](#), all matching items are selected.

If *Value* is zero or empty, the current selection is removed.

To select or deselect all items in a [multi-select `ListBox`](#), follow this example:

```
Gui.Opt("+LastFound") ; Avoids the need to
specify WinTitle below.
PostMessage(0x185, 1, -1, ListBox.ClassNN) ;
Select all items. 0x185 is LB_SETSEL.
PostMessage(0x185, 0, -1, ListBox.ClassNN) ;
Deselect all items.
ListBox.Choose(0) ; Deselect all items.
```

Unlike [Control Choose](#), this method does not raise a [Change](#) or [DoubleClick](#)

event.

## UseTab

Causes subsequently added controls to belong to the specified tab of a Tab control.

```
GuiCtrl.UseTab([Value, ExactMatch := false])
```

### Value

This parameter should be 1 for the first tab, 2 for the second, etc. If *Value* is not an integer, the tab whose leading part matches *Value* will be used. The search is not case sensitive. For example, if a the control contains the tab "UNIX Text", specifying the word unix (lowercase) would be enough to use it. If *Value* is zero, a blank string or omitted, subsequently controls are added outside the Tab control.

### ExactMatch

If this parameter is true, *Value* has to be an exact match, but not case sensitive.

# Add

Appends the specified entries at the current list of a `ListBox`, `DropDownList`, `ComboBox`, or `Tab` control.

```
GuiCtrl.Add(ArrayOrList)
```

## ArrayOrList

A pipe-delimited list of entries (e.g. `"Red|Green|Blue"`) or an array of entries (e.g. `["Red", "Green", "Blue"]`) to be appended at the end of the control's list. To replace (overwrite) the list instead, use `GuiCtrl.Delete` beforehand. When using pipes, one of the entries can be pre-selected by including two pipes after it (e.g. `"Red|Green||Blue"`), otherwise use `GuiCtrl.Choose`. The separator between text fields may be changed to something other than pipe. For example, `Gui.Opt("+Delimiter n")` would change it to linefeed and `Gui.Opt("+DelimiterTab")` would change it to tab (`^t`).

**Related:** [ListView.Add](#), [TreeView.Add](#)

## Delete

Deletes the specified entry or all entries of a ListBox, DropDownList, ComboBox, or Tab control.

```
GuiCtrl.Delete([Value])
```

### Value

This parameter should be 1 for the first entry, 2 for the second, etc. If omitted, all entries are deleted.

For tab controls, you should note that a tab's sub-controls stay associated with their original tab number; that is, they are never associated with their tab's actual display-name. For this reason, renaming or removing a tab will not change the tab number to which the sub-controls belong. For example, if there are three tabs "Red|Green|Blue" and the second tab is removed by means of `MyTab.Delete(2)`, the sub-controls originally associated with Green will now be associated with Blue. Because of this behavior, only tabs at the end should generally be removed. Tabs that are removed in this way can be added back later, at which time they will reclaim their original set of controls.

**Related:** [ListView.Delete](#), [TreeView.Delete](#)

## Value

Retrieves or sets the contents of a control.

```
RetrievedValue := GuiCtrl.Value
```

```
GuiCtrl.Value := NewValue
```

Note: If the control is not value-capable, *RetrievedValue* will be blank and assigning *NewValue* will raise an error. Invalid values throw an exception.

Following control types are value-capable:

### Picture

*RetrievedValue* is the picture's file name as it was originally specified when the control was created. This name does not change even if a new picture file name is specified.

*NewValue* is the filename (or [handle](#)) of the new image to load (see [Picture](#) for supported file types). Zero or more of the following options may be specified immediately in front of the filename: `*wN` (width N), `*hN` (height N), and `*IconN` (icon group number N in a DLL or EXE file). In the following example, the default icon from the second icon group is loaded with a width of 100 and an automatic height via "keep aspect ratio":

```
MyPic.Value("*icon2 *w100 *h-1 C:\My
Application.exe"). Specify *w0 *h0 to use the image's actual width
```

and height. If `*w` and `*h` are omitted, the image will be scaled to fit the current size of the control. When loading from a multi-icon .ICO file, specifying a width and height also determines which icon to load. Note: Use only one space or tab between the final option and the filename itself; any other spaces and tabs are treated as part of the filename.

## Text

*RetrievedValue* is the text/caption of the control.

*NewValue* is the control's new text. Since the control will not expand automatically, use `GuiCtrl.Move("w300")` if the control needs to be widened.

## Edit

*RetrievedValue* is the current content of the control. For multi-line controls, any line breaks in the text will be represented as plain linefeeds (``n`) rather than the traditional CR+LF (``r`n`) used by non-GUI commands such as `ControlGetText` and `ControlSetText`.

*NewValue* is the new content. For multi-line controls, Any linefeeds (``n`) in *NewValue* that lack a preceding carriage return (``r`) are automatically translated to CR+LF (``r`n`) to make them display properly. However, this is usually not a concern because when using `Gui.Submit()` or when retrieving this value this translation will be reversed automatically by replacing CR+LF with LF (``n`).

To retrieve or set the text without translating `\n` to or from `\r\n`, use `GuiCtrl.Text`.

### Hotkey

*RetrievedValue* is the modifiers and key name if there is a hotkey in the control; otherwise blank. Examples: `^!C`, `^Home`, `+^NumpadHome`.

*NewValue* can be a set of modifiers with a key name, or blank to clear the control. Examples: `^!c`, `^Numpad1`, `+Home`. The only modifiers supported are ^ (Control), ! (Alt), and + (Shift). See the [key list](#) for available key names.

### Checkbox / Radio

*RetrievedValue* is 1 if the control is checked, 0 if it is unchecked, or -1 if it has a gray checkmark.

*NewValue* can be 0 to uncheck the button, 1 to check it, or -1 to give it a gray checkmark. For Radio buttons, if one is being checked (turned on) and it is a member of a multi-radio group, the other radio buttons in its group will be automatically unchecked.

To get or set control's text/caption instead, use `GuiCtrl.Text`.

### DateTime / MonthCal

*RetrievedValue* is a date-time stamp in `YYYYMMDDHH24MISS` format.

*NewValue* is a date-time stamp in `YYYYMMDDHH24MISS` format. Specify `A_NOW` to use the current date and time (today). For DateTime controls,

*NewValue* may be an empty string to cause the control to have no date/time selected (if it was created with [that ability](#)). For MonthCal controls, a range may be specified if the control is [multi-select](#).

### [UpDown](#) / [Slider](#) / [Progress](#)

*RetrievedValue* is the control's current position.

*NewValue* is the new position of the control, for example

`MySlider.Value := 10`. To adjust the value by a relative amount, use the operators `+=`, `-=`, `++` or `--` instead of `:=`. If the new position would be outside the range of the control, the control is generally set to the nearest valid value.

### [Tab](#) / [DropDownList](#) / [ComboBox](#) / [ListBox](#)

*RetrievedValue* is the position of the currently selected item/tab or zero if none is selected. If text is entered into a ComboBox, the first item with matching text is used. If there is no matching item, the result is zero even if there is text in the control. For a [multi-select ListBox](#), the result is an array of numbers (which is empty if no items are selected).

*NewValue* is the position of a single item/tab to select, or zero to clear the current selection (this is valid even for Tab controls). To select multiple items, call `GuiCtrl.Choose` repeatedly.

To get or set the selected item given its text instead of its position, use `GuiCtrl.Text`.

## ActiveX

*RetrievedValue* is the ActiveX object. For example, if the control was created with the text *Shell.Explorer*, this is a [WebBrowser](#) object. The same [wrapper object](#) is returned each time.

# Text

Retrieves or sets the text/caption of the control.

```
RetrievedText := GuiCtrl.Text
```

```
GuiCtrl.Text := NewText
```

Note: If the control has no visible caption text and no (single) text value, this property corresponds to the control's hidden caption text (like [ControlGetText/ControlSetText](#)).

**Caption/display text:** The *Text* property retrieves or sets the caption/display text of the following control types: [Button](#), [Checkbox](#), [Edit](#), [GroupBox](#), [Link](#), [Radio](#), [Text](#). Since the control will not expand automatically, use `GuiCtrl.Move("w300")` or similar if the control needs to be widened.

**DateTime:** The *Text* property retrieves the formatted text displayed by the control. Assigning a formatted date/time string to the control is not supported. To change the date/time being displayed, assign [GuiCtrl.Value](#) a date-time stamp in `YYYYMMDDHH24MISS` format.

**Edit:** As with other controls, the text is retrieved or set as-is; no end-of-line translation is performed. To retrieve or set the text of a multi-line Edit control while also translating between `r`n` and ``n`, use [GuiCtrl.Value](#).

**StatusBar:** The *Text* property retrieves or sets the text of the first part only

(use `SB.SetText` for greater flexibility).

**List item text:** The *Text* property retrieves or sets the currently selected item/tab for the following control types: `Tab`, `DropDownList`, `ComboBox`, `ListBox`.

*NewText* should be the full text (case insensitive) of the item/tab to select.

For a `ComboBox`, if there is no selected item, the text in the control's edit field is retrieved instead. For other controls, *RetrievedText* is empty/blank. Similarly, if *NewText* is empty/blank, the current item/tab will be deselected.

For a `multi-select ListBox`, *RetrievedText* is an array. *NewText* cannot be an array, but if multiple items match, they are all selected. To select multiple items with different text, call `GuiCtrl.Choose` repeatedly.

To select an item by its position number instead of its text, use `GuiCtrl.Value`.

## SetFont

Sets the font typeface, size, style, and/or color for the control.

Note: Omit both parameters to set the font to the GUI's current font, as set by [Gui.SetFont](#). Otherwise, any font attributes which are not specified will be copied from the control's previous font. Text color is changed only if specified in *Options*.

```
GuiCtrl.SetFont([Options, FontName])
```

For details about both parameters, see [Gui.SetFont](#).

## Enabled

Retrieves the current interaction state of the control, or enables or disables (grays out) it.

```
RetrievedState := GuiCtrl.Enabled
```

```
GuiCtrl.Enabled := NewState
```

For Tab controls, this will also enable or disable all of the tab's sub-controls. However, any sub-control explicitly disabled via `GuiCtrl.Enabled := false` will remember that setting and thus remain disabled even after its Tab control is re-enabled.

## Visible

Retrieves the current visibility state of the control, or shows or hides it.

```
RetrievedState := GuiCtrl.Visible
```

```
GuiCtrl.Visible := NewState
```

For Tab controls, this will also show or hide all of the tab's sub-controls. If you additionally want to prevent a control's shortcut key (underlined letter) from working, disable the control via `GuiCtrl.Enabled := false`.

## Focused

Retrieves the current focus state of the control.

```
RetrievedState := GuiCtrl.Focused
```

Note: To be effective, the window generally must not be minimized or hidden.

## Pos

Retrieves the position and size of the control.

```
PosSizeObj := GuiCtrl.Pos
```

The position is relative to the GUI window's client area, which is the area not including title bar, menu bar, and borders. The information is stored in an object. For example:

```
value := MyEdit.Pos
MsgBox The X coordinate is %value.x%. The Y
coordinate is %value.y%. The width is
%value.w%. The height is %value.h%.
```

# GuiCreate

Creates a new [Gui object](#), which is essential for creating and managing a GUI.

```
GuiObj := GuiCreate([Options, Title := A_ScriptName,
EventObj])
```

## Parameters

### Options

For performance reasons, it is better to set all options in a single line.

Specify a plus sign to add the option and a minus sign to remove it. For example: `Gui.Opt("+Resize -MaximizeBox")`.

The effect of this parameter is cumulative; that is, it alters only those settings that are explicitly specified, leaving all the others unchanged.

**AlwaysOnTop:** Makes the window stay on top of all other windows, which is the same effect as [WinSetAlwaysOnTop](#).

**Border:** Provides a thin-line border around the window. This is not common.

**Caption** (present by default): Provides a title bar and a thick window border/edge. When removing the caption from a window that will use [WinSetTransparentColor](#), remove it only after setting the `TransparentColor`.

**Delimiter:** Specifies that the window should use a field separator other than pipe (|) whenever controls' contents are added via `Gui.Add` or modified via `GuiControl` object. Specify a single character immediately after the word `Delimiter`. For example, `Gui.Opt("+Delimiter`n")` would use a linefeed character, which might be especially appropriate with [continuation sections](#). Similarly, `Gui.Opt("+Delimiter|")` would revert to the default delimiter. To use space or tab, specify `Gui.Opt("+DelimiterSpace")` or `Gui.Opt("+DelimiterTab")`. Once the delimiter is changed, it affects all existing and subsequent [threads](#) that operate on this particular window.

**Disabled:** Disables the window, which prevents the user from interacting with its controls. This is often used on a window that owns other windows (see [Owner](#)).

**DPIScale:** Use `Gui.Opt("-DPIScale")` to disable DPI scaling, which is enabled by default. If DPI scaling is enabled on a system with a non-standard DPI setting, the `Gui` object and `GuiControl` object automatically scale coordinates and sizes to give controls roughly the same apparent size (but higher resolution). For example, with a DPI of 144 (150%), `E := Gui.Add("Edit", "w100")` would make the GUI control 150 pixels wide, but `E.Pos.W` would still return 100.

`A_ScreenDPI` contains the system's current DPI.

DPI scaling only applies to the `Gui` object and `GuiControl` object, so

coordinates coming directly from other sources such as `ControlGetPos` or `WinGetPos` will not work. There are a number of ways to deal with this:

- Avoid using hard-coded coordinates wherever possible. For example, use the `xp`, `xs`, `xm` and `x+m` options for positioning controls and specify height in `rows of text` instead of pixels.
- Enable (`Gui.Opt("+DPIScale")`) and disable (`Gui.Opt("-DPIScale")`) scaling on the fly, as needed. Changing the setting does not affect positions or sizes which have already been set.
- Manually scale the coordinates. For example, `x*(A_ScreenDPI/96)` converts `x` from logical/GUI coordinates to physical/non-GUI coordinates.

**LastFound:** Sets the window to be the `last found window` (though this is unnecessary in a `GUI thread` because it is done automatically), which allows functions such as `WinGetStyle` and `WinSetTransparent` to operate on it even if it is hidden (that is, `DetectHiddenWindows` is not necessary). This is especially useful for changing the properties of the window before showing it. For example:

```
Gui.Opt("+LastFound")
WinSetTransColor(CustomColor " 150")
Gui.Show()
```

**MaximizeBox:** Enables the maximize button in the title bar. This is also included as part of `Resize` below.

**MinimizeBox** (present by default): Enables the minimize button in the title bar.

**MinSize** and **MaxSize**: Determines the minimum and/or maximum size of the window, such as when the user drags its edges to resize it. Specify the word *MinSize* and/or *MaxSize* with no suffix to use the window's current size as the limit (if the window has no current size, it will use the size from the first use of `Gui.Show`). Alternatively, append the width, followed by an X, followed by the height; for example:

`Gui.Opt("+Resize +MinSize640x480")`. The dimensions are in pixels, and they specify the size of the window's client area (which excludes borders, title bar, and `menu bar`). Specify each number as decimal, not hexadecimal.

Either the width or the height may be omitted to leave it unchanged (e.g. `+MinSize640x` or `+MinSizex480`). Furthermore, `Min/MaxSize` can be specified more than once to use the window's current size for one dimension and an explicit size for the other. For example, `+MinSize +MinSize640x` would use the window's current size for the height and 640 for the width.

If *MinSize* and *MaxSize* are never used, the operating system's defaults are used (similarly, `Gui.Opt("-MinSize -MaxSize")` can be used to return to the defaults). Note: the window must have `+Resize` to allow resizing by the user.

**OwnDialogs**: `Gui.Opt("+OwnDialogs")` should be specified in

each [thread](#) (such as a event handling function of a Button control) for which subsequently displayed [MsgBox](#), [InputBox](#), [FileSelect](#), and [DirSelect](#) dialogs should be owned by the window. Such dialogs are modal, meaning that the user cannot interact with the GUI window until dismissing the dialog. By contrast, [ToolTip](#) do not become modal even though they become owned; they will merely stay always on top of their owner. In either case, any owned dialog or window is automatically destroyed when its GUI window is [destroyed](#).

There is typically no need to turn this setting back off because it does not affect other [threads](#). However, if a thread needs to display both owned and unowned dialogs, it may turn off this setting via `Gui.Opt("-OwnDialogs")`.

**Owner:** Use `+Owner` to make the window owned by another. An owned window has no taskbar button by default, and when visible it is always on top of its owner. It is also automatically destroyed when its owner is destroyed. `+Owner` can be used before or after the owned window is created. There are two ways to use `+Owner`, as shown below:

```
Gui.Opt("+OwnerMyOtherGui") ; Make the GUI
owned by MyOtherGui.
Gui.Opt("+Owner") ; Make the GUI owned by
script's main window to prevent display of
a taskbar button.
```

`+Owner` can be immediately followed by the [name](#) or number of an existing GUI or the [HWND](#) of any top-level window.

To prevent the user from interacting with the owner while one of its owned window is visible, disable the owner via `Gui.Opt("+Disabled")`. Later (when the time comes to cancel or destroy the owned window), re-enable the owner via `Gui.Opt("-Disabled")`. Do this prior to cancel/destroy so that the owner will be reactivated automatically.

**Parent:** Use `+Parent` immediately followed by the `name` or number of an existing GUI or the `HWND` of any window or control to use it as the parent of this window. To convert the GUI back into a top-level window, use `-Parent`. This option works even after the window is created.

**Resize:** Makes the window resizable and enables its maximize button in the title bar. To avoid enabling the maximize button, specify `+Resize` `-MaximizeBox`.

**SysMenu** (present by default): Specify `-SysMenu` (minus SysMenu) to omit the system menu and icon in the window's upper left corner. This will also omit the minimize, maximize, and close buttons in the title bar.

**Theme:** By specifying `-Theme`, all subsequently created controls in the window will have Classic Theme appearance on Windows XP and beyond. To later create additional controls that obey the current theme, turn it back on via `+Theme`. Note: This option has no effect on operating systems older than Windows XP, nor does it have any effect on XP itself if the Classic Theme is in effect. Finally, this setting may be changed for an individual control by specifying `+Theme` or `-Theme` in its options

when it is created.

**ToolWindow:** Provides a narrower title bar but the window will have no taskbar button.

**(Unnamed Style):** Specify a plus or minus sign followed immediately by a decimal or hexadecimal [style number](#).

**(Unnamed ExStyle):** Specify a plus or minus sign followed immediately by the letter E and a decimal or hexadecimal extended style number. For example, `+E0x40000` would add the `WS_EX_APPWINDOW` style, which provides a taskbar button for a window that would otherwise lack one. Although the other extended styles are not documented here (since they are rarely used), they can be discovered by searching for `WS_EX_APPWINDOW` at [www.microsoft.com](http://www.microsoft.com).

## Title

The window title. If omitted, it defaults to the current value of [A\\_ScriptName](#).

## EventObj

An "event sink", or object to bind events to. If *EventObj* is specified, [OnEvent](#), [OnNotify](#) and [OnCommand](#) can be used to register methods of *EventObj* to be called when an event is raised. If omitted or empty, any string passed to [OnEvent](#)'s *Function* parameter is interpreted as a function name.

## Return Value

Returns a [Gui object](#). This object provides methods and properties for creating and managing windows, and creating controls.

## Keyboard Navigation

A GUI window may be navigated via the TAB key, which moves keyboard focus to the next input-capable control (controls from which the [Tabstop](#) style has been removed are skipped). The order of navigation is determined by the order in which the controls were originally added. When the window is shown for the first time, the first input-capable control that has the Tabstop style (which most control types have by default) will have keyboard focus.

Certain controls may contain an ampersand (&) to create a keyboard shortcut, which might be displayed in the control's text as an underlined character (depending on system settings). A user activates the shortcut by holding down the ALT key then typing the corresponding character. For buttons, checkboxes, and radio buttons, pressing the shortcut is the same as clicking the control. For GroupBoxes and Text controls, pressing the shortcut causes keyboard focus to jump to the first input-capable [tabstop](#) control that was created after it. However, if more than one control has the same shortcut key, pressing the shortcut will alternate keyboard focus among all controls with the same shortcut.

To display a literal ampersand inside the control types mentioned above, specify two consecutive ampersands as in this example: `Save && Exit`.

## Window Appearance

For its icon, a GUI window uses the tray icon that was in effect at the time the window was created. Thus, to have a different icon, change the tray icon before creating the window. For example: `Menu("Tray", "Icon", "MyIcon.ico")`. It is also possible to have a different large icon for a window than its small icon (the large icon is displayed in the alt-tab task switcher). This can be done via `DllCall` and `SendMessage`; for example:

```
hIcon32 := DllCall("LoadImage", uint, 0
, str, "My Icon.ico" ; Icon filename (this
file may contain multiple icons).
, uint, 1 ; Type of image: IMAGE_ICON
, int, 32, int, 32 ; Desired width and
height of image (helps LoadImage decide which
icon is best).
, uint, 0x10) ; Flags: LR_LOADFROMFILE
Gui := GuiCreate("+LastFound")
SendMessage(0x80, 1, hIcon32) ; 0x80 is
WM_SETICON; and 1 means ICON_BIG (vs. 0 for
ICON_SMALL).
Gui.Show()
```

Due to OS limitations, Checkboxes, Radio buttons, and GroupBoxes for which a non-default text color was specified will take on Classic Theme appearance on Windows XP and beyond.

Related topic: [window's margin](#).

## General Remarks

Use the [GuiControl](#) object to operate upon individual controls in a GUI window.

Each GUI window may have up to 11,000 controls. However, use caution when creating more than 5000 controls because system instability may occur for certain control types.

If the script is not [persistent](#) for any other reason, it will exit after the last visible GUI is closed; either when the last thread completes or immediately if no threads are running.

## Related

[Gui](#) object, [GuiControl](#) object, [GuiFromHwnd](#), [GuiCtrlFromHwnd](#), [Menu](#), [Control Types](#), [ListView](#), [TreeView](#), [Control functions](#), [MsgBox](#), [FileSelect](#), [DirSelect](#)

## Examples

```
; Example: Display a text pop-up:
```

```
Gui := GuiCreate(, "Title of Window")
Gui.Opt("+AlwaysOnTop +Disabled -SysMenu +Owner")
; +Owner avoids a taskbar button.
Gui.Add("Text",, "Some text to display.")
Gui.Show("NoActivate") ; NoActivate avoids
deactivating the currently active window.
```

```
; Example: A simple input-box that asks for first
name and last name:
```

```

Gui := GuiCreate(, "Simple Input Example")
Gui.Add("Text",, "First name:")
Gui.Add("Text",, "Last name:")
Gui.Add("Edit", "vFirstName ym") ; The ym option
starts a new column of controls.
Gui.Add("Edit", "vLastName")
Gui.Add("Button", "default",
"OK").OnEvent("Click", "Gui_Close")
Gui.OnEvent("Close", "Gui_Close")
Gui.Show()

Gui_Close(Gui)
{
 Saved := Gui.Submit() ; Save the contents of
named controls into an object.
 MsgBox("You entered '%Saved.FirstName%
%Saved.LastName%'")
}

```

### **; Example: Tab control:**

```

Gui := GuiCreate()
Tab := Gui.Add("Tab3",, "First Tab|Second
Tab|Third Tab")
Gui.Add("Checkbox", "vMyCheckbox", "Sample
checkbox")
Tab.UseTab(2)
Gui.Add("Radio", "vMyRadio", "Sample radio1")
Gui.Add("Radio",, "Sample radio2")
Tab.UseTab(3)
Gui.Add("Edit", "vMyEdit r5") ; r5 means 5 rows
tall.
Tab.UseTab() ; i.e. subsequently-added controls
will not belong to the tab control.
Gui.Add("Button", "default xm",
"OK").OnEvent("Click", "Gui_Close") ; xm puts it

```

at the bottom left corner.

```
Gui.OnEvent("Close", "Gui_Close")
Gui.OnEvent("Escape", "Gui_Close")
Gui.Show()
```

```
Gui_Close(Gui)
```

```
{
```

```
 Saved := Gui.Submit() ; Save the contents of
 named controls into an object.
```

```
 MsgBox("You entered:`n" Saved.MyCheckbox "`n"
 Saved.MyRadio "`n" Saved.MyEdit)
```

```
}
```

; Example: ListBox containing files in a directory:

```
Gui := GuiCreate()
```

```
Gui.Add("Text",, "Pick a file to launch from the
list below.")
```

```
fn := Func("LaunchFile").bind(Gui)
```

```
LB := Gui.Add("ListBox", "w640 r10 vFile"),
```

```
LB.OnEvent("DoubleClick", fn)
```

```
LoopFiles("C:*.*)" ; Change this folder and
wildcard pattern to suit your preferences.
```

```
 LB.Add(A_LoopFilePath)
```

```
Gui.Add("Button", "Default",
"OK").OnEvent("Click", fn)
```

```
Gui.Show()
```

```
LaunchFile(Gui)
```

```
{
```

```
 Saved := Gui.Submit()
```

```
 if MsgBox("Would you like to launch the file or
document below?`n`n%Saved.File%",, 4) = "No"
```

```
 return
```

```
 try ; Otherwise, try to launch it:
```

```
 Run(Saved.File)
 catch Error
 MsgBox(Error.Extra)
}
```

**; Example: Display context-sensitive help (via ToolTip) whenever the user moves the mouse over a particular control:**

```
Gui := GuiCreate()
Gui.Add("Edit", "vMyEdit")
MyEdit_TT := "This is a tooltip for the control
whose name is MyEdit."
Gui.Add("DropDownList", "vMyDDL",
"Red|Green|Blue")
MyDDL_TT := "Choose a color from the drop-down
list."
Gui.Add("Checkbox", "vMyCheckBox", "This control
has no tooltip.")
Gui.OnEvent("Close", "Gui_Close")
Gui.Show()
OnMessage(0x200, "WM_MOUSEMOVE")

WM_MOUSEMOVE(wParam, lParam, msg, Hwnd)
{
 global
 static PrevHwnd
 if (Hwnd <gt; PrevHwnd)
 {
 Text := "", ToolTip() ; Turn off any previous
 tooltip.
 CurrControl := GuiCtrlFromHwnd(Hwnd)
 if CurrControl
 {
 Text := %CurrControl.Name%_TT
 SetTimer("DisplayToolTip", -1000)
 }
 }
}
```

```

 }
 PrevHwnd := Hwnd
 }
}

DisplayToolTip()
{
 global
 ToolTip(Text)
 SetTimer("ToolTip", -3000) ; Remove the tooltip.
}

Gui_Close()
{
 ExitApp()
}

```

**; Example: On-screen display (OSD) via transparent window:**

```

Gui := GuiCreate()
Gui.Opt("+LastFound +AlwaysOnTop -Caption
+ToolWindow") ; +ToolWindow avoids a taskbar
button and an alt-tab menu item.
Gui.BackColor := "EEAA99" ; Can be any RGB color
(it will be made transparent below).
Gui.SetFont("s32") ; Set a large font size (32-
point).
CoordText := Gui.Add("Text", "cLime", "XXXXX
YYYYY") ; XX & YY serve to auto-size the window.
; Make all pixels of this color transparent and
make the text itself translucent (150):
WinSetTransColor(Gui.BackColor " 150")
fn := Func("UpdateOSD").bind(CoordText),
SetTimer(fn, 200)
UpdateOSD(CoordText) ; Make the first update

```

immediate rather than waiting for the timer.  
Gui.Show("x0 y400 NoActivate") ; NoActivate  
avoids deactivating the currently active window.

```
UpdateOSD(GuiCtrl)
{
 MouseGetPos(MouseX, MouseY)
 GuiCtrl.Value := "X%MouseX%, Y%MouseY%"
}
```

; Example: A moving progress bar overlaid on a  
background image.

```
Gui := GuiCreate()
Gui.BackColor := "White"
Gui.Add("Picture", "x0 y0 h350 w450",
"%A_WinDir%\Web\Wallpaper\Windows\img0.jpg")
MyBtn := Gui.Add("Button", "Default xp+20 yp+250",
"Start the Bar Moving")
MyProgress := Gui.Add("Progress", "w416")
MyText := Gui.Add("Text", "wp") ; wp means "use
width of previous".
MyBtn.OnEvent("Click",
Func("MoveBar").bind(MyProgress, MyText))
Gui.Show()

MoveBar(ProgressCtrl, TextCtrl)
{
 LoopFiles("%A_WinDir%*.*)"
 {
 if A_Index > 100
 break
 ProgressCtrl.Value := A_Index
 TextCtrl.Value := A_LoopFileName
 Sleep 50
 }
}
```

```
 TextCtrl.Value := "Bar finished."
}
```

### **; Example: Simple image viewer:**

```
Gui := GuiCreate("+Resize")
MyBtn := Gui.Add("Button", "default", "&Load; New
Image")
MyRadio := Gui.Add("Radio", "ym+5 x+10 checked",
"Load &actual; size")
Gui.Add("Radio", "ym+5 x+10", "Load to &fit;
screen")
MyPic := Gui.Add("Pic", "xm vPic")
MyBtn.OnEvent("Click",
Func("LoadNewImage").bind(Gui, MyRadio, MyPic))
Gui.Show()
```

```
LoadNewImage(Gui, RadioCtrl, PicCtrl)
{
 File := FileSelect(, "Select an image:",
"Images (*.gif; *.jpg; *.bmp; *.png; *.tif; *.ico;
*.cur; *.ani; *.exe; *.dll)")
 if File = ""
 return
 if RadioCtrl.Value = 1 ; Display image at its
actual size.
 {
 Width := 0
 Height := 0
 }
 else ; Second radio is selected: Resize the
image to fit the screen.
 {
 Width := A_ScreenWidth - 28 ; Minus 28 to
allow room for borders and margins inside.
 Height := -1 ; "Keep aspect ratio" seems
```

```

best.
}
PicCtrl.Value := "*w%width% *h%Height% %File%"
; Load the image.
Gui.Title := File
Gui.Show("xCenter y0 AutoSize") ; Resize the
window to match the picture size.
}

```

```

; Example: Simple text editor with menu bar.

```

```

; Make these variables accessible for all the
functions below:

```

```

global Gui, MainEdit, CurrentFileName, About

```

```

; Create the GUI window:

```

```

Gui := GuiCreate("+Resize", "Untitled") ; Make
the window resizable.

```

```

; Create the sub-menus for the menu bar:

```

```

Menu("FileMenu", "Add", "&New;", "MenuFileNew")
Menu("FileMenu", "Add", "&Open;", "MenuFileOpen")
Menu("FileMenu", "Add", "&Save;", "MenuFileSave")
Menu("FileMenu", "Add", "Save &As;",
"MenuFileSaveAs")
Menu("FileMenu", "Add") ; Separator line.
Menu("FileMenu", "Add", "E&xit;", "MenuFileExit")
Menu("HelpMenu", "Add", "&About;",
"MenuHelpAbout")

```

```

; Create the menu bar by attaching the sub-menus
to it:

```

```

Menu("MyMenuBar", "Add", "&File;", ":FileMenu")
Menu("MyMenuBar", "Add", "&Help;", ":HelpMenu")

```

```

; Attach the menu bar to the window:

```

```

Gui.Menu := "MyMenuBar"

; Create the main Edit control:
MainEdit := Gui.Add("Edit", "WantTab W600 R20")

; Apply events:
Gui.OnEvent("DropFiles", "Gui_DropFiles")
Gui.OnEvent("Size", "Gui_Size")

MenuFileNew() ; Apply default settings.
Gui.Show() ; Display the window.

MenuFileNew()
{
 MainEdit.Value := "" ; Clear the Edit control.
 Menu("FileMenu", "Disable", "3&") ; Gray out
 &Save.;
 Gui.Title := "Untitled"
}

MenuFileOpen()
{
 Gui.Opt("+OwnDialogs") ; Force the user to
 dismiss the FileSelect dialog before returning to
 the main window.
 SelectedFileName := FileSelect(3,, "Open File",
 "Text Documents (*.txt)")
 if SelectedFileName = "" ; No file selected.
 return
 CurrentFileName := readContent(SelectedFileName)
}

MenuFileSave()
{
 saveContent(CurrentFileName)
}

```

```

MenuFileSaveAs()
{
 Gui.Opt("+OwnDialogs") ; Force the user to
dismiss the FileSelect dialog before returning to
the main window.
 SelectedFileName := FileSelect("S16",, "Save
File, Text Documents (*.txt)")
 if SelectedFileName = "" ; No file selected.
 return
 CurrentFileName := saveContent(SelectedFileName)
}

MenuFileExit() ; User chose "Exit" from the File
menu.
{
 WinClose()
}

MenuHelpAbout()
{
 About := GuiCreate("+owner%Gui.Hwnd%") ; Make
the main window the owner of the "about box".
 Gui.Opt("+Disabled") ; Disable main window.
 About.Add("Text",, "Text for about box.")
 About.Add("Button", "Default",
"OK").OnEvent("Click", "About_Close")
 About.OnEvent("Close", "About_Close")
 About.OnEvent("Escape", "About_Close")
 About.Show()
}

readContent(FileName)
{
 FileContent := FileRead(FileName) ; Read the
file's contents into the variable.
 if ErrorLevel
 {

```

```

 MsgBox("Could not open '%FileName%'.")
 return
}
MainEdit.Value := FileContent ; Put the text
into the control.
Menu("FileMenu", "Enable", "3&") ; Re-enable
&Save.;
Gui.Title := FileName ; Show file name in title
bar.
return FileName
}

saveContent(FileName)
{
 if FileExist(FileName)
 {
 FileDelete(FileName)
 if ErrorLevel
 {
 MsgBox("The attempt to overwrite
'%FileName%' failed.")
 return
 }
 }
}
FileAppend(MainEdit.Value, FileName) ; Save the
contents to the file.
; Upon success, Show file name in title bar (in
case we were called by MenuFileSaveAs):
Gui.Title := FileName
return FileName
}

About_Close()
{
 Gui.Opt("-Disabled") ; Re-enable the main
window (must be done prior to the next step).
 About.Destroy() ; Destroy the about box.
}

```

```
}

Gui_DropFiles(Gui, FileArray) ; Support drag &
drop.
{
 CurrentFileName := readContent(FileArray[1]) ;
 Read the first file only (in case there's more
 than one).
}

Gui_Size(Gui, MinMax, Width, Height)
{
 if MinMax = -1 ; The window has been minimized.
 No action needed.
 return
 ; Otherwise, the window has been resized or
 maximized. Resize the Edit control to match.
 MainEdit.Move("W%Width-20% H%Height-20%")
}
```

# GuiCtrlFromHwnd

Retrieves the [GuiControl object](#) of a GUI control associated with the specified HWND.

```
GuiObj := GuiCtrlFromHwnd(Hwnd)
```

## Parameters

### Hwnd

The window handle (HWND) of a GUI control. Such control could be created with [Gui.Add](#).

## Return Value

Returns a [GuiControl object](#). This object provides methods and properties for modifying GUI controls and retrieving information about them.

## Remarks

For example, a HWND of a GUI control can be retrieved via [GuiCtrl.Hwnd](#), [MouseGetPos](#) or [OnMessage](#).

## Related

[GuiCreate](#), [Gui object](#), [GuiControl object](#), [GuiFromHwnd](#), [Menu](#), [Control Types](#), [ListView](#), [TreeView](#), [Control functions](#), [MsgBox](#), [FileSelect](#), [DirSelect](#)

## Examples

See the [ToolTip example](#) on the GuiCreate page.

# GuiFromHwnd

Retrieves the [Gui](#) object of a GUI window associated with the specified HWND.

```
GuiObj := GuiFromHwnd(Hwnd [, RecurseParent := false])
```

## Parameters

### Hwnd

The window handle (HWND) of a GUI window. Such window could be created with [GuiCreate](#).

### RecurseParent

If this parameter is true, the closest parent to the specified HWND that is a GUI is automatically searched for and retrieved.

## Return Value

Returns a [Gui](#) object. This object provides methods and properties for creating and managing windows, and creating controls.

## Remarks

For example, a HWND of a GUI window can be retrieved via [Gui.Hwnd](#), [WinExist](#) or [WinGet](#).

## Related

GuiCreate, Gui object, GuiControl object, GuiCtrFromHwnd, Menu, Control Types, ListView, TreeView, Control functions, MsgBox, FileSelect, DirSelect

## Examples

```
Gui := GuiCreate(, "Title of Window")
Gui.Add("Text",, "Some text to display.")
Gui.Show()

MsgBox(GuiFromHwnd(Gui.Hwnd).Title)
```

# List View

## Table of Contents

- [Introduction and Simple Example](#)
- [Options and Styles](#)
- [View Modes: Report \(default\), Icon, Tile, Small-Icon, and List.](#)
- [Built-in Methods:](#)
  - [Row methods \(adding, modifying, and deleting rows\)](#)
  - [Column methods](#)
  - [Getting data out of a ListView](#)
  - [Setting icons](#)
- [Events](#)
- [ImageLists \(the means by which icons are added to a ListView\)](#)
- [ListView Remarks](#)
- [Examples](#)

## Introduction and Simple Example

A List-View is one of the most elaborate controls provided by the operating system. In its most recognizable form, it displays a tabular view of rows and columns, the most common example of which is Explorer's list of files and folders (detail view).

Though it may be elaborate, a ListView's basic features are easy to use. The syntax for creating a ListView is:

```
LV := Gui.Add("ListView", Options,
"ColumnName1|ColumnName2|...")
```

Here is a working script that creates and displays a ListView containing a list of files in the user's "My Documents" folder:

```
; Create the window.
Gui := GuiCreate()

; Create the ListView with two columns, Name
and Size:
LV := Gui.Add("ListView", "r20 w700",
"Name|Size (KB)")

; Notify the script whenever the user double
clicks a row:
LV.OnEvent("DoubleClick", "LV_DoubleClick")

; Gather a list of file names from a folder and
put them into the ListView:
```

```
LoopFiles("%A_MyDocuments%*.*")
 LV.Add(, A_LoopFileName, A_LoopFileSizeKB)

LV.ModifyCol() ; Auto-size each column to fit
its contents.
LV.ModifyCol(2, "Integer") ; For sorting
purposes, indicate that column 2 is an integer.

; Display the window.
Gui.Show()

LV_DoubleClick(LV, RowNumber)
{
 RowText := LV.GetText(RowNumber) ; Get the
text from the row's first field.
 ToolTip("You double-clicked row number
%RowNumber%. Text: '%RowText%'")
}
```

## Options and Styles for the *Options* parameter

**Background:** Specify the word `Background` followed immediately by a color name (see [color chart](#)) or RGB value (the `0x` prefix is optional). Examples: `BackgroundSilver`, `BackgroundFFDD99`. If this option is not present, the `ListView` initially defaults to the system's default background color. Specifying `BackgroundDefault` or `-Background` applies the system's default background color (usually white). For example, a `ListView` can be restored to the default color via `LV.Opt("+BackgroundDefault")`.

**C:** Text color. Specify the letter `C` followed immediately by a color name (see [color chart](#)) or RGB value (the `0x` prefix is optional). Examples: `cRed`, `cFF2211`, `c0xFF2211`, `cDefault`.

**Checked:** Provides a checkbox at the left side of each row. When [adding](#) a row, specify the word `Check` in its options to have the box to start off checked instead of unchecked. The user may either click the checkbox or press the spacebar to check or uncheck a row.

**Count:** Specify the word `Count` followed immediately by the total number of rows that the `ListView` will ultimately contain. This is not a limit: rows beyond the count can still be added. Instead, this option serves as a hint to the control that allows it to allocate memory only once rather than each time a row is added, which greatly improves row-adding performance (it may also improve sorting performance). To improve performance even more, use `LV.Opt("-Redraw")` prior to adding a large number of rows. Afterward, use

`LV.Opt("+Redraw")` to re-enable redrawing (which also repaints the control).

**Grid:** Provides horizontal and vertical lines to visually indicate the boundaries between rows and columns.

**Hdr:** Specify `-Hdr` (minus Hdr) to omit the special top row that contains column titles. To make it visible later, use `LV.Opt("+Hdr")`.

**LV:** Specify the string LV followed immediately by the number of an [extended ListView style](#). These styles are entirely separate from generic extended styles. For example, specifying `-E0x200` would remove the generic extended style `WS_EX_CLIENTEDGE` to eliminate the control's default border. By contrast, specifying `-LV0x20` would remove `LVS_EX_FULLROWSELECT`.

**LV0x10:** Specify `-LV0x10` to prevent the user from dragging column headers to the left or right to reorder them. However, it is usually not necessary to do this because the physical reordering of columns does not affect the column order seen by the script. For example, the first column will always be column 1 from the script's point of view, even if the user has physically moved it to the right of other columns.

**LV0x20:** Specify `-LV0x20` to require that a row be clicked at its first field to select it (normally, a click on *any* field will select it). The advantage of this is that it makes it easier for the user to drag a rectangle around a group of rows to select them.

**Multi:** Specify `-Multi` (minus Multi) to prevent the user from selecting more

than one row at a time.

**NoSortHdr:** Prevents the header from being clickable. It will take on a flat appearance rather than its normal button-like appearance. Unlike most other ListView styles, this one cannot be changed after the ListView is created.

**NoSort:** Turns off the automatic sorting that occurs when the user clicks a column header. However, the header will still behave visually like a button (unless NoSortHdr has been specified). In addition, the [ColClick](#) event is still raised, so the script can respond with a custom sort or other action.

**ReadOnly:** Specify `-ReadOnly` (minus ReadOnly) to allow editing of the text in the first column of each row. To edit a row, select it then press the [F2 key](#). Alternatively, you can click a row once to select it, wait at least half a second, then click the same row again to edit it.

**R:** Rows of height (upon creation). Specify the letter R followed immediately by the number of rows for which to make room inside the control. For example, `R10` would make the control 10 rows tall. If the ListView is created with a [view mode](#) other than report view, the control is sized to fit rows of icons instead of rows of text. Note: adding [icons](#) to a ListView's rows will increase the height of each row, which will make this option inaccurate.

**Sort:** The control is kept alphabetically sorted according to the contents of the first column.

**SortDesc:** Same as above except in descending order.

**WantF2:** Specify `-WantF2` (minus `WantF2`) to prevent an F2 keystroke from editing the currently focused row. This setting is ignored unless `-ReadOnly` is also in effect.

**(Unnamed numeric styles):** Since styles other than the above are rarely used, they do not have names. See the [ListView styles table](#) for a list.

## View Modes

A `ListView` has five viewing modes, of which the most common is report view (which is the default). To use one of the other views, specify its name in the options list. The view can also be changed after the control is created; for example: `LV.Opt("+IconSmall")`.

**Icon:** Shows a large-icon view. In this view and all the others except *Report*, the text in columns other than the first is not visible. To display icons in this mode, the `ListView` must have a large-icon `ImageList` assigned to it.

**Tile:** Shows a large-icon view but with ergonomic differences such as displaying each item's text to the right of the icon rather than underneath it. `Checkboxes` do not function in this view. Also, attempting to show this view on operating systems older than Windows XP has no effect.

**IconSmall:** Shows a small-icon view.

**List:** Shows a small-icon view in list format, which displays the icons in columns. The number of columns depends on the width of the control and the width of the widest text item in it.

**Report:** Switches back to report view, which is the initial default. For example: `LV.Opt("+Report")`.

## Built-in Methods for ListViews

Additionally to the [default methods/properties of a GUI control](#), the following methods can be specified for a ListView. If the associated [GuiControl object](#) does not exist or does not belongs to a ListView, the ListView methods throw an [exception](#).

When the phrase "row number" is used on this page, it refers to a row's current position within the ListView. The top row is 1, the second row is 2, and so on. After a row is added, its row number tends to change due to sorting, deleting, and inserting of other rows. Therefore, to locate specific row(s) based on their contents, it is usually best to use [LV.GetText](#) in a loop.

## Row Methods

### Add

Adds a new row to the bottom of the list, and returns the new [row number](#), which is not necessarily the last row if the ListView has the [Sort](#) or [SortDesc](#) style.

```
NewRowNumber := LV.Add([Options, Col1, Col2, ...])
```

### Options

A string containing zero or more words from the list below (not case sensitive). Separate each word from the next with a space or tab. To

remove an option, precede it with a minus sign. To add an option, a plus sign is permitted but not required.

**Check:** Shows a checkmark in the row (if the ListView has checkboxes). To later uncheck it, use `LV.Modify(RowNumber, "-Check")`.

**Col:** Specify the word Col followed immediately by the column number at which to begin applying the parameters *Col1* and beyond. This is most commonly used with `LV.Modify` to alter individual fields in a row without affecting those that lie to their left.

**Focus:** Sets keyboard focus to the row (often used in conjunction with Select). To later de-focus it, use `LV.Modify(RowNumber, "-Focus")`.

**Select:** Selects the row. To later deselect it, use `LV.Modify(RowNumber, "-Select")`. When selecting rows, it is usually best to ensure that at least one row always has the `focus` property because that allows the Apps key to display its context menu (if any) near the focused row. The word *Select* may optionally be followed immediately by a 0 or 1 to indicate the starting state. In other words, both `"Select"` and `"Select".VarContainingOne` are the same (the period used here is the concatenation operator). This technique also works with *Focus* and *Check* above.

**Vis:** Ensures that the specified row is completely visible by scrolling

the ListView, if necessary. This has an effect only for LV.Modify; for example: `LV.Modify(RowNumber, "Vis")`.

### Col1, Col2, ...

The columns of the new row, which can be text or numeric (including numeric [expression](#) results). To make any field blank, specify "" or the equivalent. If there are too few fields to fill all the columns, the columns at the end are left blank. If there are too many fields, the fields at the end are completely ignored.

## Insert

Inserts a new row at the specified row number, and returns the new [row number](#).

```
NewRowNumber := LV.Insert(RowNumber [, Options,
Col1, Col2, ...])
```

### RowNumber

The row number for the newly inserted row. Any rows at or beneath *RowNumber* are shifted downward to make room for the new row. If *RowNumber* is greater than the number of rows in the list (even as high as 2147483647), the new row is added to the end of the list.

### Options

See [row options](#).

## Col1, Col2, ...

The columns of the new row, which can be text or numeric (including numeric [expression](#) results). To make any field blank, specify "" or the equivalent. If there are too few fields to fill all the columns, the columns at the end are left blank. If there are too many fields, the fields at the end are completely ignored.

## Modify

Modifies the attributes and/or text of a row, and returns 1 upon success and 0 upon failure.

```
LV.Modify(LineNumber [, Options, NewCol1, NewCol2,
...])
```

Note: When only the first two parameters are present, only the row's attributes and not its text are changed.

## LineNumber

The row to modify. If *LineNumber* is 0, all rows in the control are modified (in this case the return value is 1 on complete success and 0 if any part of the operation failed).

## Options

The [ColN option](#) may be used to update specific columns without affecting the others. For other options, see [row options](#).

### NewCol1, NewCol2, ...

The new columns of the specified row, which can be text or numeric (including numeric [expression](#) results). To make any field blank, specify "" or the equivalent. If there are too few parameters to cover all the columns, the columns at the end are not changed. If there are too many fields, the fields at the end are completely ignored.

## Delete

Deletes the specified row, and returns 1 upon success and 0 upon failure.

```
LV.Delete([RowNumber])
```

### RowNumber

The row to delete. If this parameter is omitted, **all** rows in the ListView are deleted.

## Column Methods

### ModifyCol

Modifies the attributes and/or text of the specified column and its header, and returns 1 upon success and 0 upon failure.

```
LV.ModifyCol([ColumnNumber, Options, ColumnTitle])
```

Note: If all parameters are omitted, the width of every column is adjusted to fit the contents of the rows. If only the first parameter is present, only the specified column is auto-sized. Auto-sizing has no effect when not in Report (Details) view.

### ColumnNumber

The column to modify. The first column is number 1 (not 0).

### Options

A string containing zero or more words from the list below (not case sensitive). Separate each word from the next with a space or tab. To remove an option, precede it with a minus sign. To add an option, a plus sign is permitted but not required.

#### Column Options: General

**N:** Specify for N the new width of the column, in pixels. This number can be unquoted if is the only option. For example, the following are both valid: `LV.ModifyCol(1, 50)` and `LV.ModifyCol(1, "50 Integer")`.

**Auto:** Adjusts the column's width to fit its contents. This has no effect when not in Report (Details) view.

**AutoHdr:** Adjusts the column's width to fit its contents and the column's header text, whichever is wider. If applied to the last column, it will be made at least as wide as all the remaining space in the

ListView. It is usually best to apply this setting only after the rows have been added because that allows any newly-arrived vertical scrollbar to be taken into account when sizing the last column. This option has no effect when not in Report (Details) view.

**Icon:** Specify the word Icon followed immediately by the number of the `ImageList`'s icon to display next to the column header's text.

Specify `-Icon` (minus icon) to remove any existing icon.

**IconRight:** Puts the icon on the right side of the column rather than the left.

### Column Options: Data Type

**Float:** For sorting purposes, indicates that this column contains floating point numbers (hexadecimal format is not supported). Sorting performance for Float and Text columns is up to 25 times slower than it is for integers.

**Integer:** For sorting purposes, indicates that this column contains integers. To be sorted properly, each integer must be 32-bit; that is, within the range -2147483648 to 2147483647. If any of the values are not integers, they will be considered zero when sorting (unless they start with a number, in which case that number is used). Numbers may appear in either decimal or hexadecimal format (e.g. `0xF9E0`).

**Text:** Changes the column back to text-mode sorting, which is the initial default for every column. Only the first 8190 characters of text

are significant for sorting purposes (except for the *Logical* option, in which case the limit is 4094).

### **Column Options: Alignment / Justification**

**Center:** Centers the text in the column. To center an Integer or Float column, specify the word Center after the word Integer or Float.

**Left:** Left-justifies the column's text, which is the initial default for every column. On older operating systems, the first column might have a forced left-justification.

**Right:** Right-justifies the column's text. This attribute need not be specified for Integer and Float columns because they are right-justified by default. That default can be overridden by specifying something such as `"Integer Left"` or `"Float Center"`.

### **Column Options: Sorting**

**Case:** The sorting of the column is case sensitive (affects only `text` columns). If the options *Case*, *CaseLocale*, and *Logical* are all omitted, the uppercase letters A-Z are considered identical to their lowercase counterparts for the purpose of the sort.

**CaseLocale:** The sorting of the column is case insensitive based on the current user's locale (affects only `text` columns). For example, most English and Western European locales treat the letters A-Z and ANSI letters like Ä and Ü as identical to their lowercase counterparts. This

method also uses a "word sort", which treats hyphens and apostrophes in such a way that words like "coop" and "co-op" stay together.

**Desc:** Descending order. The column starts off in descending order the first time the user sorts it.

**Logical:** Same as *CaseLocale* except that any sequences of digits in the text are treated as true numbers rather than mere characters. For example, the string "T33" would be considered greater than "T4". *Logical* requires Windows XP or later (on older OSes, *CaseLocale* is automatically used instead). In addition, *Logical* and *Case* are currently mutually exclusive: only the one most recently specified will be in effect.

**NoSort:** Prevents a user's click on this column from having any automatic sorting effect. However, the [ColClick](#) event is still raised, so the script can respond with a custom sort or other action. To disable sorting for all columns rather than only a subset, include [NoSort](#) in the `ListView`'s options.

**Sort:** Immediately sorts the column in ascending order (even if it has the [Desc](#) option).

**SortDesc:** Immediately sorts the column in descending order.

**Uni:** Unidirectional sort. This prevents a second click on the same column from reversing the sort direction.

## ColumnTitle

The new column header. Omit this parameter to left it unchanged.

## InsertCol

Inserts a new column at the specified column number, and returns the new column's position number.

```
NewColumnNumber := LV.InsertCol(ColumnNumber [, Options, ColumnTitle])
```

## ColumnNumber

The column number of the newly inserted column. Any column at or on the right side of *ColumnNumber* are shifted to the right to make room for the new column. The first column is 1 (not 0). The maximum number of columns in a *ListView* is 200. If *ColumnNumber* is larger than the number of columns currently in the control, the new column is added next to the last column on the right side. The newly inserted column starts off with empty contents beneath it unless it is the first column, in which case it inherits the old first column's contents and the old first column acquires blank contents.

## Options

The new column's attributes -- such as whether or not it uses [integer sorting](#) -- always start off at their defaults unless changed via *Options*.

### ColumnTitle

The new column header.

## DeleteCol

Deletes the specified column and all of the contents beneath it, and returns 1 upon success and 0 upon failure.

```
LV.DeleteCol(ColumnNumber)
```

### ColumnNumber

The column to delete. Once a column is deleted, the column numbers of any that lie to its right are reduced by 1. Consequently, calling `LV.DeleteCol(2)` twice would delete the second and third columns. On operating systems older than Windows XP, attempting to delete the original first column might fail and return 0.

## Getting Data Out of a ListView

### GetCount

Returns the number of rows or columns in the control.

```
CountNumber := LV.GetCount([Mode])
```

## Mode

When this parameter is omitted, it returns the total number of rows in the control. When this parameter is "S" or "Selected", the count includes only the selected/highlighted rows. When this parameter is "Col" or "Column", it returns the number of columns in the control. The return value will be retrieved instantly, because the control keeps track of these counts.

This method is often used in the top line of a Loop, in which case the method would get called only once (prior to the first iteration). For example:

```
Loop(LV.GetCount())
{
 RetrievedText := LV.GetText(A_Index)
 if InStr(RetrievedText, "some filter
text")
 LV.Modify(A_Index, "Select") ; Select
each row whose first field contains the
filter-text.
}
```

To retrieve the widths of a ListView's columns -- for uses such as saving them to an INI file to be remembered between sessions -- follow this example:

```
Gui.Opt("+LastFound")
Loop(LV.GetCount("Column"))
{
 ColWidth := SendMessage(4125, A_Index - 1,
0, "SysListView321") ; 4125 is
LVM_GETCOLUMNWIDTH.
 MsgBox("Column %A_Index%'s width is
```

```
%ColWidth%.")
}
```

## GetNext

Returns the row number of the next selected, checked, or focused row, otherwise zero.

```
RowNumber := LV.GetNext([StartingRowNumber,
RowType])
```

### StartingRowNumber

If omitted or less than 1, the search begins at the top of the list.  
Otherwise, the search begins at the row after *StartingRowNumber*.

### RowType

If omitted, it searches for the next selected/highlighted row. Otherwise, specify "C" or "Checked" to find the next checked row; or "F" or "Focused" to find the focused row (there is never more than one focused row in the entire list, and sometimes there is none at all).

The following example reports all selected rows in the ListView:

```
RowNumber := 0 ; This causes the first loop
iteration to start the search at the top of
the list.
Loop
{
```

```

 RowNumber := LV.GetNext(RowNumber) ;
Resume the search at the row after that found
by the previous iteration.
 if not RowNumber ; The above returned
zero, so there are no more selected rows.
 break
 Text := LV.GetText(RowNumber)
 MsgBox('The next selected row is
#%RowNumber%, whose first field is "%Text%".')
}

```

An alternate method to find out if a particular row number is checked is the following:

```

Gui.Opt("+LastFound")
ItemState := SendMessage(4140, RowNumber - 1,
0xF000, "SysListView321") ; 4140 is
LVM_GETITEMSTATE. 0xF000 is
LVIS_STATEIMAGEMASK.
IsChecked := (ItemState >> 12) - 1 ; This
sets IsChecked to true if RowNumber is checked
or false otherwise.

```

## GetText

Retrieves the text at the specified row and column number.

```

RetrievedText := LV.GetText(RowNumber [,
ColumnNumber])

```

## RowNumber

The number of the row, whose text to be retrieved. If this parameter is 0, the column header text is retrieved. *RetrievedText* has a maximum length of 8191.

## ColumnNumber

The number of the column, where the specified row is located. If omitted, it defaults to 1 (the text in the first column). Column numbers seen by the script are not altered by any dragging and dropping of columns the user may have done. For example, the original first column is still number 1 even if the user drags it to the right of other columns.

## Setting Icons

### SetImageList

Sets or replaces the [ImageList](#), and returns the ImageListID that was previously associated with this control (or 0 if none).

```
PrevImageListID := LV.SetImageList(ImageListID [,
IconType])
```

## ImageListID

The number returned from a previous call to [IL\\_Create](#).

## IconType

If the second parameter is omitted, the type of icons in the ImageList is detected automatically as large or small. Otherwise, specify 0 for large icons, 1 for small icons, and 2 for state icons (state icons are not yet directly supported, but they could be used via [SendMessage](#)).

This method is normally called prior to adding any rows to the ListView.

A ListView may have up to two ImageLists: small-icon and/or large-icon.

This is useful when the script allows the user to switch to and from the large-icon view. To add more than one ImageList to a ListView, call

`LV.SetImageList` a second time, specifying the ImageListID of the second list.

A ListView with both a large-icon and small-icon ImageList should ensure that both lists contain the icons in the same order. This is because the same ID number is used to reference both the large and small versions of a particular icon.

Although it is traditional for all [viewing modes](#) except Icon and Tile to show small icons, this can be overridden by passing a large-icon list to `LV.SetImageList` and specifying 1 (small-icon) for the second parameter. This also increases the height of each row in the ListView to fit the large icon.

Any such detached ImageList should normally be destroyed via [IL\\_Destroy](#).

## Events

The following events can be detected by calling [OnEvent](#) to register a callback function or method:

| Event                       | Raised when...                                                                                                   |
|-----------------------------|------------------------------------------------------------------------------------------------------------------|
| <a href="#">Click</a>       | The control is clicked.                                                                                          |
| <a href="#">DoubleClick</a> | The control is double-clicked.                                                                                   |
| <a href="#">ColClick</a>    | A column header is clicked.                                                                                      |
| <a href="#">ContextMenu</a> | The user right-clicks the control or presses the Apps key or Shift-F10 while the control has the keyboard focus. |
| <a href="#">Focus</a>       | The control gains the keyboard focus.                                                                            |
| <a href="#">LoseFocus</a>   | The control loses the keyboard focus.                                                                            |
| <a href="#">ItemCheck</a>   | An item is checked or unchecked.                                                                                 |
| <a href="#">ItemEdit</a>    | An item's label is edited by the user.                                                                           |
| <a href="#">ItemFocus</a>   | The focused item changes.                                                                                        |
| <a href="#">ItemSelect</a>  | An item is selected or deselected.                                                                               |

Additional (rarely-used) notifications can be detected by using [OnNotify](#). These notifications are [documented at MSDN](#). MSDN does not show the numeric value of each notification code; those can be found in the Windows SDK or by searching the Internet.

## ImageList (the means by which icons are added to a ListView)

An Image-List is a group of identically sized icons stored in memory. Upon creation, each ImageList is empty. The script calls `IL_Add` repeatedly to add icons to the list, and each icon is assigned a sequential number starting at 1. This is the number to which the script refers to display a particular icon in a row or column header. Here is a working example that demonstrates how to put icons into a ListView's rows:

```
Gui := GuiCreate() ; Create a GUI window.
LV := Gui.Add("ListView", "h200 w180", "Icon &
Number|Description") ; Create a ListView.
ImageListID := IL_Create(10) ; Create an
ImageList to hold 10 small icons.
LV.SetImageList(ImageListID) ; Assign the
above ImageList to the current ListView.
Loop(10) ; Load the ImageList with a series of
icons from the DLL.
 IL_Add(ImageListID, "shell32.dll", A_Index)
Loop(10) ; Add rows to the ListView (for
demonstration purposes, one for each icon).
 LV.Add("Icon" . A_Index, A_Index, "n/a")
LV.ModifyCol("Hdr") ; Auto-adjust the column
widths.
Gui.Show()
```

## IL\_Create

Creates a new ImageList, initially empty, and returns the unique ID of the ImageList (or 0 upon failure).

```
UniqueID := IL_Create([InitialCount := 2, GrowCount := 5, LargeIcons := false])
```

### InitialCount

The number of icons you expect to put into the list immediately (if omitted, it defaults to 2).

### GrowCount

The number of icons by which the list will grow each time it exceeds the current list capacity (if omitted, it defaults to 5).

### LargeIcons

If this parameter is true, the ImageList will contain large icons. If it is false, it will contain small icons (this is the default when omitted). Icons added to the list are scaled automatically to conform to the system's dimensions for small and large icons.

## IL\_Add

Adds an icon or picture to the specified ImageList, and returns the new icon's index (1 is the first icon, 2 is the second, and so on).

```
IconIndex := IL_Add(ImageListID, Filename [, IconNumber := 1, ResizeNonIcon])
```

### ImageListID

The ID of a ImageList created by `IL_Create`.

## Filename

The name of an icon (.ICO), cursor (.CUR), animated cursor (.ANI) file (animated cursors will not actually be animated when displayed in a ListView), or a [bitmap or icon handle](#) like `HBITMAP:%handle%`. Other sources of icons include the following types of files: EXE, DLL, CPL, SCR, and other types that contain icon resources.

## IconNumber

To use an icon group other than the first one in the file, specify its number for *IconNumber*. If *IconNumber* is negative, its absolute value is assumed to be the resource ID of an icon within an executable file. In the following example, the default icon from the second icon group would be used:

```
IL_Add(ImageListID, "C:\My Application.exe", 2).
```

## ResizeNonIcon

Non-icon images such as BMP, GIF and JPG may also be loaded. However, in this case the last two parameters should be specified to ensure correct behavior: *IconNumber* should be the mask/transparency color number (0xFFFFFFFF [the color white] might be best for most pictures); and *ResizeNonIcon* should be non-zero to cause the picture to be scaled to become a single icon, or zero to divide up the image into however many icons can fit into its actual width.

All operating systems support GIF, JPG, BMP, ICO, CUR, and ANI images. On Windows XP or later, additional image formats such as PNG, TIF, Exif, WMF, and EMF are supported. Operating systems older than

XP can be given support by copying Microsoft's free GDI+ DLL into the AutoHotkey.exe folder (but in the case of a [compiled script](#), copy the DLL into the script's folder). To download the DLL, search for the following phrase at [www.microsoft.com](http://www.microsoft.com): gdi redistributable

## IL\_Destroy

Deletes the specified ImageList, and returns 1 upon success and 0 upon failure.

```
Success := IL_Destroy(ImageListID)
```

### ImageListID

The ID of a ImageList created by [IL\\_Create](#).

Note: It is normally not necessary to destroy ImageLists because once attached to a ListView, they are destroyed automatically when the ListView or its parent window is destroyed. However, if the ListView shares ImageLists with other ListViews (by having `0x40` in its options), the script should explicitly destroy the ImageList after destroying all the ListViews that use it. Similarly, if the script replaces one of a ListView's old ImageLists with a new one, it should explicitly destroy the old one.

## Listview Remarks

`Gui.Submit` has no effect on a `Listview` control.

After a column is sorted -- either by means of the user clicking its header or the script calling `LV.ModifyCol(1, "Sort")` -- any subsequently added rows will appear at the bottom of the list rather than obeying the sort order. The exception to this is the `Sort` and `SortDesc` styles, which move newly added rows into the correct positions.

To detect when the user has pressed Enter while a `Listview` has focus, use a `default button` (which can be hidden if desired). For example:

```
Gui.Add("Button", "Hidden Default",
"OK").OnEvent("Click", "LV_Enter")
...
LV_Enter() {
 global
 if Gui.FocusedCtrl != LV
 return
 MsgBox("Enter was pressed. The focused row
number is " LV.GetNext(0, "Focused"))
}
```

In addition to navigating from row to row with the keyboard, the user may also perform incremental search by typing the first few characters of an item in the first column. This causes the selection to jump to the nearest matching row.

Although any length of text can be stored in each field of a `Listview`, only the

first 260 characters are displayed.

Although the maximum number of rows in a `ListView` is limited only by available system memory, row-adding performance can be greatly improved as described in the `Count` option.

A picture may be used as a background around a `ListView` (that is, to frame the `ListView`). To do this, create the `picture control` after the `ListView` and include `0x40000000` (which is `WS_CLIPSIBLINGS`) in the picture's `Options`.

A script may create more than one `ListView` per window.

It is best not to insert or delete columns directly with `SendMessage`. This is because the program maintains a collection of `sorting preferences` for each column, which would then get out of sync. Instead, use the `built-in column methods`.

To perform actions such as resizing, hiding, or changing the font of a `ListView`, see `GuiControl` object.

To extract text from external `ListView`s (those not owned by the script), use `ControlGetList`.

## Related

[TreeView](#), [Other Control Types](#), [GuiCreate](#), [ContextMenu event](#), [Gui object](#), [GuiControl object](#), [ListView styles table](#)

## Examples

```
; Select or de-select all rows by specifying 0 as
the row number:
```

```
LV.Modify(0, "Select") ; Select all.
LV.Modify(0, "-Select") ; De-select all.
LV.Modify(0, "-Check") ; Uncheck all the
checkboxes.
```

```
; Auto-size all columns to fit their contents:
```

```
LV.ModifyCol() ; There are no parameters in this
mode.
```

```
; MAIN EXAMPLE
```

```
; The following is a working script that is more
elaborate than the one near the top of this page.
```

```
; It displays the files in a folder chosen by the
user, with each file assigned the icon associated
with
```

```
; its type. The user can double-click a file, or
right-click one or more files to display a context
menu.
```

```
; Create a GUI window:
```

```
Gui := GuiCreate("+Resize") ; Allow the user to
maximize or drag-resize the window.
```

```
; Create some buttons:
```

```
B_1 := Gui.Add("Button", "Default", "Load a
folder")
B_2 := Gui.Add("Button", "x+20", "Clear List")
B_3 := Gui.Add("Button", "x+20", "Switch View")
```

**; Create the ListView and its columns:**

```
LV := Gui.Add("ListView", "xm r20 w700", "Name|In
Folder|Size (KB)|Type")
```

**LV.ModifyCol(3, "Integer") ; For sorting,  
indicate that the Size column is an integer.**

**; Apply ListView events:**

```
LV.OnEvent("DoubleClick", "RunFile")
```

```
LV.OnEvent("ContextMenu", "ShowContextMenu")
```

**; Create an ImageList so that the ListView can  
display some icons:**

```
ImageListID1 := IL_Create(10)
```

**ImageListID2 := IL\_Create(10, 10, true) ; A list  
of large icons to go with the small ones.**

**; Attach the ImageLists to the ListView so that it  
can later display the icons:**

```
LV.SetImageList(ImageListID1)
```

```
LV.SetImageList(ImageListID2)
```

**; Apply control events:**

```
LV.OnEvent("DoubleClick", "RunFile")
```

```
LV.OnEvent("ContextMenu", "ShowContextMenu")
```

```
B_1.OnEvent("Click", Func("LoadFolder").bind(Gui,
LV, ImageListID1, ImageListID2))
```

```
B_2.OnEvent("Click", Func("ClearList").bind(LV))
```

```
B_3.OnEvent("Click", Func("SwitchView").bind(LV))
```

**; Apply window events:**

```
Gui.OnEvent("Size", Func("Gui_Size").bind(LV))
```

**; Display the window:**

```
Gui.Show()
```

```
LoadFolder(Gui, LV, ImageListID1, ImageListID2)
```

```

{
 static IconArray := []
 Gui.Opt("+OwnDialogs") ; Forces user to dismiss
the following dialog before using main window.
 Folder := DirSelect(, 3, "Select a folder to
read:")
 if not Folder ; The user canceled the dialog.
 return

 ; Check if the last character of the folder name
is a backslash, which happens for root
 ; directories such as C:\. If it is, remove it
to prevent a double-backslash later on.
 if SubStr(Folder, -1, 1) = "\"
 Folder := SubStr(Folder, 1, -1) ; Remove the
trailing backslash.

 ; Calculate buffer size required for SHFILEINFO
structure.
 sfi_size := A_PtrSize + 688
 VarSetCapacity(sfi, sfi_size)

 ; Gather a list of file names from the selected
folder and append them to the ListView:
 LV.Opt("-Redraw") ; Improve performance by
disabling redrawing during load.
 LoopFiles("%Folder%*.*)")
 {
 FileName := A_LoopFilePath ; Must save it to
a writable variable for use below.

 ; Build a unique extension ID to avoid
characters that are illegal in variable names,
 ; such as dashes. This unique ID method also
performs better because finding an item
 ; in the array does not require search-loop.
 SplitPath(FileName,,, FileExt) ; Get the

```

file's extension.

```
if FileExt ~= "(EXE|ICO|ANI|CUR)$"
{
 ExtID := FileExt ; Special ID as a
placeholder.
 IconNumber := 0 ; Flag it as not found so
that these types can each have a unique icon.
}
else ; Some other extension/file-type, so
calculate its unique ID.
{
 ExtID := 0 ; Initialize to handle
extensions that are shorter than others.
 Loop(7) ; Limit the extension to 7
characters so that it fits in a 64-bit value.
{
 ExtChar := SubStr(FileExt, A_Index, 1)
 if not ExtChar ; No more characters.
 break
 ; Derive a Unique ID by assigning a
different bit position to each character:
 ExtID := ExtID | (Ord(ExtChar) << (8 *
(A_Index - 1)))
}
 ; Check if this file extension already has
an icon in the ImageLists. If it does,
 ; several calls can be avoided and loading
performance is greatly improved,
 ; especially for a folder containing
hundreds of files:
 IconNumber := IconArray[ExtID]
}
if not IconNumber ; There is not yet any icon
for this extension, so load it.
{
 ; Get the high-quality small-icon associated
with this file extension:
```

```

 if not DllCall("Shell32\SHGetFileInfo",
"str", FileName
 , "uint", 0, "ptr", &sfi, "uint", sfi_size,
"uint", 0x101) ; 0x101 is
SHGFI_ICON+SHGFI_SMALLICON
 IconNumber := 9999999 ; Set it out of
bounds to display a blank icon.
 else ; Icon successfully loaded.
 {
 ; Extract the hIcon member from the
structure:
 hIcon := NumGet(sfi, 0)
 ; Add the HICON directly to the small-icon
and large-icon lists.
 ; Below uses +1 to convert the returned
index from zero-based to one-based:
 IconNumber :=
DllCall("ImageList_ReplaceIcon", "ptr",
ImageListID1, "int", -1, "ptr", hIcon) + 1
 DllCall("ImageList_ReplaceIcon", "ptr",
ImageListID2, "int", -1, "ptr", hIcon)
 ; Now that it's been copied into the
ImageLists, the original should be destroyed:
 DllCall("DestroyIcon", "ptr", hIcon)
 ; Cache the icon to save memory and
improve loading performance:
 IconArray[ExtID] := IconNumber
 }
}

; Create the new row in the ListView and
assign it the icon number determined above:
LV.Add("Icon" . IconNumber, A_LoopFileName,
A_LoopFileDir, A_LoopFileSizeKB, FileExt)
}
LV.Opt("+Redraw") ; Re-enable redrawing (it was
disabled above).

```

```

 LV.ModifyCol() ; Auto-size each column to fit
its contents.
 LV.ModifyCol(3, 60) ; Make the Size column at
little wider to reveal its header.
}

ClearList(LV)
{
 LV.Delete() ; Clear the ListView, but keep icon
cache intact for simplicity.
}

SwitchView(LV)
{
 static IconView
 if not IconView
 LV.Opt("+Icon") ; Switch to icon view.
 else
 LV.Opt("+Report") ; Switch back to
details view.
 IconView := not IconView ; Invert in
preparation for next time.
}

RunFile(LV, RowNumber, Verb := "")
{
 FileName := LV.GetText(RowNumber, 1) ; Get the
text of the first field.
 FileDir := LV.GetText(RowNumber, 2) ; Get the
text of the second field.
 FilePath := FileDir "\ " FileName
 Verb .= Verb ? " " : ""
 try
 Run(Verb FilePath)
 catch Error
 MsgBox(Error.Extra "`n" FilePath)
}

```

```

ShowContextMenu(LV, Item, IsRightClick, X, Y) ;
In response to right-click or Apps key.
{
 ; Create function references for the menu
 events:
 ContextOpen := Func("ContextOpen").bind(LV,
Item)
 ContextProperties :=
Func("ContextProperties").bind(LV, Item)
 ContextClearRows :=
Func("ContextClearRows").bind(LV)

 ; Create or update the popup menu to be used as
 the context menu:
 Menu("MyContextMenu", "Add", "Open",
ContextOpen)
 Menu("MyContextMenu", "Add", "Properties",
ContextProperties)
 Menu("MyContextMenu", "Add", "Clear from
ListView", ContextClearRows)
 Menu("MyContextMenu", "Default", "Open") ; Make
"Open" a bold font to indicate that double-click
does the same thing.
 ; Show the menu at the provided coordinates, X
and Y. These should be used
 ; because they provide correct coordinates even
if the user pressed the Apps key:
 Menu("MyContextMenu", "Show", X, Y)
}

ContextOpen(LV, RowNumber) ; The user selected
"Open" in the context menu.
{
 RunFile(LV, RowNumber)
}

```

```

ContextProperties(LV, RowNumber) ; The user
selected "Properties" in the context menu.
{
 RunFile(LV, RowNumber, "properties")
}

ContextClearRows(LV) ; The user selected "Clear"
in the context menu.
{
 RowNumber := 0 ; This causes the first
iteration to start the search at the top.
 Loop
 {
 ; Since deleting a row reduces the RowNumber
of all other rows beneath it,
 ; subtract 1 so that the search includes the
same row number that was previously
 ; found (in case adjacent rows are selected):
 RowNumber := LV.GetNext(RowNumber - 1)
 if not RowNumber ; The above returned zero,
so there are no more selected rows.
 break
 LV.Delete(RowNumber) ; Clear the row from the
ListView.
 }
}

Gui_Size(LV, Gui, MinMax, Width, Height) ;
Expand/Shrink ListView in response to the user's
resizing.
{
 if MinMax = -1 ; The window has been minimized.
No action needed.
 return
 ; Otherwise, the window has been resized or
maximized. Resize the ListView to match.
 LV.Move("W" Width-20 " H" Height-40)
}

```



# TreeView

## Table of Contents

- [Introduction and Simple Example](#)
- [Options and Styles](#)
- [Built-in Methods:](#)
  - [Add/modify/delete items](#)
  - [Getting data out of a TreeView](#)
  - [Setting Icons](#)
- [Events](#)
- [Remarks](#)
- [Longer Example](#)

## Introduction and Simple Example

A Tree-View displays a hierarchy of items by indenting child items beneath their parents. The most common example is Explorer's tree of drives and folders.

The syntax for creating a TreeView is:

```
TV := Gui.Add("TreeView", Options)
```

Here is a working script that creates and displays a simple hierarchy of items:

```
Gui := GuiCreate()
TV := Gui.Add("TreeView")
P1 := TV.Add("First parent")
P1C1 := TV.Add("Parent 1's first child", P1) ;
Specify P1 to be this item's parent.
P2 := TV.Add("Second parent")
P2C1 := TV.Add("Parent 2's first child", P2)
P2C2 := TV.Add("Parent 2's second child", P2)
P2C2C1 := TV.Add("Child 2's first child", P2C2)

Gui.Show() ; Show the window and its TreeView.
```

## Options and Styles for the *Options* parameter

**Background:** Specify the word `Background` followed immediately by a color name (see [color chart](#)) or RGB value (the `0x` prefix is optional). Examples: `BackgroundSilver`, `BackgroundFFDD99`. If this option is not present, the `TreeView` initially defaults to the system's default background color. Specifying `BackgroundDefault` or `-Background` applies the system's default background color (usually white). For example, a `TreeView` can be restored to the default color via `TV.Opt("+BackgroundDefault")`.

**Buttons:** Specify `-Buttons` (minus Buttons) to avoid displaying a plus or minus button to the left of each item that has children.

**C:** Text color. Specify the letter `C` followed immediately by a color name (see [color chart](#)) or RGB value (the `0x` prefix is optional). Examples: `cRed`, `cFF2211`, `c0xFF2211`, `cDefault`.

**Checked:** Provides a checkbox at the left side of each item. When adding an item, specify the word *Check* in its options to have the box to start off checked instead of unchecked. The user may either click the checkbox or press the spacebar to check or uncheck an item. To discover which items in a `TreeView` are currently checked, call `TV.GetNext` or `TV.Get`.

**HScroll:** Specify `-HScroll` (minus HScroll) to disable horizontal scrolling in the control (in addition, the control will not display any horizontal scroll bar).

**ImageList:** This is the means by which icons are added to a `TreeView`. Specify

the word *ImageList* followed immediately by the ImageListID returned from a previous call to `IL_Create`. This option has an effect only when creating a `TreeView` (however, `TV.SetImageList` does not have this limitation). Here is a working example:

```
Gui := GuiCreate()
ImageListID := IL_Create(10) ; Create an
ImageList with initial capacity for 10 icons.
Loop 10 ; Load the ImageList with some
standard system icons.
 IL_Add(ImageListID, "shell32.dll", A_Index)
TV := Gui.Add("TreeView",
"ImageList%ImageListID%")
TV.Add("Name of Item", 0, "Icon4") ; Add an
item to the TreeView and give it a folder icon.
Gui.Show()
```

**Lines:** Specify `-Lines` (minus Lines) to avoid displaying a network of lines connecting parent items to their children. However, removing these lines also prevents the plus/minus buttons from being shown for top-level items.

**ReadOnly:** Specify `-ReadOnly` (minus ReadOnly) to allow editing of the text/name of each item. To edit an item, select it then press the `F2` key. Alternatively, you can click an item once to select it, wait at least half a second, then click the same item again to edit it. After being edited, an item can be alphabetically repositioned among its siblings via the following example:

```
TV := Gui.Add("TreeView", "-ReadOnly")
TV.OnEvent("ItemEdit", "MyTree_Edit") ; Call
MyTree_Edit whenever a user has finished
```

```
editing an item.
; ...
MyTree_Edit(TV, Item)
{
 TV.Modify(TV.GetParent(Item), "Sort");
This works even if the item has no parent.
}
```

**R:** Rows of height (upon creation). Specify the letter R followed immediately by the number of rows for which to make room inside the control. For example, **R10** would make the control 10 items tall.

**WantF2:** Specify **-wantF2** (minus WantF2) to prevent an F2 keystroke from editing the currently selected item. This setting is ignored unless **-ReadOnly** is also in effect.

**(Unnamed numeric styles):** Since styles other than the above are rarely used, they do not have names. See the [TreeView styles table](#) for a list.

## Built-in Methods for TreeViews

Additionally to the [default methods/properties of a GUI control](#), the following methods can be specified for a TreeView. If the associated [GuiControl object](#) does not exist or does not belongs to a TreeView, the TreeView methods throw an [exception](#).

### Add, Modify, and Delete Items

#### Add

Adds a new item to the TreeView, and returns its unique Item ID number.

```
UniqueID := TV.Add(Name, [ParentItemID, Options])
```

#### Name

The displayed text of the item, which can be text or numeric (including numeric [expression](#) results).

#### ParentItemID

The ID number of the new item's parent (omit it or specify 0 to add the item at the top level).

#### Options

Contains zero or more words from the list below (not case sensitive). Separate each word from the next with a space or tab. To remove an

option, precede it with a minus sign. To add an option, a plus sign is permitted but not required.

**Bold:** Displays the item's name in a bold font. To later un-bold the item, use `TV.Modify(ItemID, "-Bold")`.

**Check:** Shows a checkmark to the left of the item (if the TreeView has checkboxes). To later uncheck it, use `TV.Modify(ItemID, "-Check")`. The word *Check* may optionally be followed immediately by a 0 or 1 to indicate the starting state. In other words, both `"Check"` and `"Check".VarContainingOne` are the same (the period used here is the concatenation operator).

**Expand:** Expands the item to reveal its children (if any). To later collapse the item, use `TV.Modify(ItemID, "-Expand")`. If there are no children, `TV.Modify` returns 0 instead of the item's ID. By contrast, `TV.Add` marks the item as expanded in case children are added to it later. Unlike "Select" (below), expanding an item does not automatically expand its parent. Finally, the word *Expand* may optionally be followed immediately by a 0 or 1 to indicate the starting state. In other words, both `"Expand"` and `"Expand".VarContainingOne` are the same.

**First | Sort | N:** These options apply only to `TV.Add`. They specify the new item's position relative to its siblings (a *sibling* is any other item on the same level). If none of these options is present, the new item is added as the last/bottom sibling. Otherwise, specify *First* to add the

item as the first/top sibling, or specify *Sort* to insert it among its siblings in alphabetical order. If a plain integer (**N**) is specified, it is assumed to be ID number of the sibling after which to insert the new item (if integer N is the only option present, it does not have to be enclosed in quotes).

**Icon:** Specify the word *Icon* followed immediately by the number of this item's icon, which is displayed to the left of the item's name. If this option is absent, the first icon in the [ImageList](#) is used. To display a blank icon, specify a number that is larger than the number of icons in the [ImageList](#). If the control lacks an [ImageList](#), no icon is displayed nor is any space reserved for one.

**Select:** Selects the item. Since only one item at a time can be selected, any previously selected item is automatically de-selected. In addition, this option reveals the newly selected item by expanding its parent(s), if necessary. To find out the current selection, call [TV.GetSelection](#).

**Sort:** For [TV.Modify](#), this option alphabetically sorts the children of the specified item. To instead sort all top-level items, use `TV.Modify(0, "Sort")`. If there are no children, 0 is returned instead of the ID of the modified item.

**Vis:** Ensures that the item is completely visible by scrolling the [TreeView](#) and/or expanding its parent, if necessary.

**VisFirst:** Same as above except that the [TreeView](#) is also scrolled so

that the item appears at the top, if possible. This option is typically more effective when used with `TV.Modify` than with `TV.Add`.

Note: When adding a large number of items, performance can be improved by using `TV.Opt("-Redraw")` before adding the items, and `TV.Opt("+Redraw")` afterward.

## Modify

Modifies the attributes and/or name of an item, and returns the item's own ID.

```
CurrentItemID := TV.Modify(ItemID [, Options, NewName])
```

### ItemID

The item to modify. When only this parameter is present, the specified item is [selected](#).

### Options

See the list above.

### NewName

The new name of the item. If omitted, the current name is left unchanged.

## Delete

Deletes the specified item, and returns 1 upon success and 0 upon failure.

```
TV.Delete([ItemID])
```

### ItemID

The item to delete. If omitted, **all** items in the TreeView are deleted.

## Getting Data Out of a TreeView

### GetSelection

Returns the selected item's ID number.

```
SelectedItemID := TV.GetSelection()
```

### GetCount

Returns the total number of items in the control.

```
CountNumber := TV.GetCount()
```

Note: The return value will be retrieved instantly, because the control keeps

track of the count.

## GetParent

Returns the specified item's parent as an item ID.

```
ParentItemID := TV.GetParent(ItemID)
```

### ItemID

The item to check. Items at the top level have no parent and thus return 0.

## GetChild

Returns the ID number of the specified item's first/top child (or 0 if none).

```
TopChildID := TV.GetChild(ParentItemID)
```

### ParentItemID

The parent item to check.

## GetPrev

Returns the ID number of the sibling above the specified item (or 0 if none).

```
PrevItemID := TV.GetPrev(ItemID)
```

### ItemID

The item to check.

## GetNext

Returns the ID number of the next item below the specified item (or 0 if none).

```
NextItemID := TV.GetNext([ItemID := 0, ItemType :=
""])
```

### ItemID

The item to check. If this parameter is 0 or omitted, the ID number of the first/top item in the TreeView is returned.

### ItemType

If omitted, the ID number of the sibling below the specified item will be retrieved. Otherwise specify "Full" or "F" to retrieve the next item regardless of its relationship to the specified item; or specify "Check", "Checked", or "C" to get only the next item with a checkmark.

The following example traverse the entire tree, item by item:

```
ItemID := 0 ; Causes the loop's first
iteration to start the search at the top of
```

the tree.

```
Loop
{
 ItemID := TV.GetNext(ItemID, "Full") ;
 Replace "Full" with "Checked" to find all
 checkmarked items.
 if not ItemID ; No more items in tree.
 break
 ItemText := TV.GetText(ItemID)
 MsgBox('The next Item is %ItemID%, whose
 text is "%ItemText%'.')
}
```

## GetText

Retrieves the text/name of the specified item.

```
RetrievedText := TV.GetText(ItemID)
```

### ItemID

The ID number of the item, whose text to be retrieved. *RetrievedText* has a maximum length of 8191.

## Get

Returns a non-zero value (item ID) if the specified item has the specified attribute.

```
CurrentItemID := TV.Get(ItemID, Attribute)
```

### ItemID

The item to check.

### Attribute

Specify "E", "Expand", or "Expanded" to determine if the item is currently **expanded** (that is, its children are being displayed); specify "C", "Check", or "Checked" to determine if the item has a **checkmark**; or specify "B" or "Bold" to determine if the item is currently **bold** in font.

**Tip:** Since an IF-statement sees any non-zero value as "true", **if**

```
TV.Get(ItemID, "Checked") = ItemID
```

 and **if**

```
TV.Get(ItemID, "Checked")
```

 are functionally identical.

## Setting Icons

### SetImageList

Sets or replaces the **ImageList**, and returns the ImageListID that was previously associated with this control (or 0 if none).

```
PrevImageListID := TV.SetImageList(ImageListID ,
IconType)
```

## ImageListID

The number returned from a previous call to `IL_Create`.

## IconType

If omitted, it defaults to 0. Otherwise, specify 2 for state icons (state icons are not yet directly supported, but they could be used via `SendMessage`).

Any such detached ImageList should normally be destroyed via `IL_Destroy`.

## Events

The following events can be detected by calling [OnEvent](#) to register a callback function or method:

| Event                       | Raised when...                                                                                                   |
|-----------------------------|------------------------------------------------------------------------------------------------------------------|
| <a href="#">Click</a>       | The control is clicked.                                                                                          |
| <a href="#">DoubleClick</a> | The control is double-clicked.                                                                                   |
| <a href="#">ContextMenu</a> | The user right-clicks the control or presses the Apps key or Shift-F10 while the control has the keyboard focus. |
| <a href="#">Focus</a>       | The control gains the keyboard focus.                                                                            |
| <a href="#">LoseFocus</a>   | The control loses the keyboard focus.                                                                            |
| <a href="#">ItemCheck</a>   | An item is checked or unchecked.                                                                                 |
| <a href="#">ItemEdit</a>    | An item's label is edited by the user.                                                                           |
| <a href="#">ItemExpand</a>  | An item is expanded or collapsed.                                                                                |
| <a href="#">ItemSelect</a>  | An item is selected.                                                                                             |

Additional (rarely-used) notifications can be detected by using [OnNotify](#). These notifications are [documented at MSDN](#). MSDN does not show the numeric value of each notification code; those can be found in the Windows SDK or by searching the Internet.

## Remarks

To detect when the user has pressed Enter while a TreeView has focus, use a [default button](#) (which can be hidden if desired). For example:

```
Gui.Add("Button", "Hidden Default",
"OK").OnEvent("Click", "ButtonOK")
...
ButtonOK() {
 global
 if Gui.FocusedCtrl != TV
 return
 MsgBox("Enter was pressed. The selected item
ID is " TV.GetSelection())
}
```

In addition to navigating from item to item with the keyboard, the user may also perform incremental search by typing the first few characters of an item's name. This causes the selection to jump to the nearest matching item.

Although any length of text can be stored in each item of a TreeView, only the first 260 characters are displayed.

Although the theoretical maximum number of items in a TreeView is 65536, item-adding performance will noticeably decrease long before then. This can be alleviated somewhat by using the redraw tip described in [TV.Add](#).

Unlike [ListViews](#), a TreeView's ImageList is not automatically destroyed when the TreeView is destroyed. Therefore, a script should call [IL\\_Destroy](#) after

destroying a TreeView's window if the ImageList will not be used for anything else. However, this is not necessary if the script will soon be exiting because all ImageLists are automatically destroyed at that time.

A script may create more than one TreeView per window.

To perform actions such as resizing, hiding, or changing the font of a TreeView, see [GuiControl object](#).

Tree View eXtension (TVX) extends TreeViews to support moving, inserting and deleting. It is demonstrated at [www.autohotkey.com/forum/topic19021.html](http://www.autohotkey.com/forum/topic19021.html)

## Related

[ListView](#), [Other Control Types](#), [GuiCreate](#), [ContextMenu event](#), [Gui object](#), [GuiControl object](#), [TreeView styles table](#)

## Example

```
; The following is a working script that is more elaborate than the one near the top of this page.
; It creates and displays a TreeView containing all folders of the user's document folder. When the
the
; user selects a folder, its contents are shown in a ListView to the right (like Windows Explorer).
; In addition, a StatusBar control shows information about the currently selected folder.
```

```
; The following folder will be the root folder for the TreeView. Note that loading might take a long
; time if an entire drive such as C:\ is specified:
```

```
TreeRoot := A_MyDocuments
TreeViewWidth := 280
ListViewWidth := A_ScreenWidth/2 - TreeViewWidth - 30
```

```
; Create the GUI window and display the source directory (TreeRoot) in the title bar:
Gui := GuiCreate("+Resize", TreeRoot) ; Allow the user to maximize or drag-resize the window.
```

```
; Create an ImageList and put some standard system icons into it:
```

```
ImageListID := IL_Create(5)
Loop 5
 IL_Add(ImageListID, "shell32.dll", A_Index)
; Create a TreeView and a ListView side-by-side to behave like Windows Explorer:
TV := Gui.Add("TreeView", "r20 w%TreeViewWidth% ImageList%ImageListID%")
```

```
LV := Gui.Add("ListView", "r20 w%ListViewWidth%
x+10", "Name|Modified")
```

```
; Create a Status Bar to give info about the
number of files and their total size:
```

```
SB := Gui.Add("StatusBar")
SB.SetParts(60, 85) ; Create three parts in the
bar (the third part fills all the remaining
width).
```

```
; Add folders and their subfolders to the tree.
Display the status in case loading takes a long
time:
```

```
M := GuiCreate("ToolWindow -SysMenu", "Loading the
tree.."), M.Show()
DirList := AddSubFoldersToTree(TV, TreeRoot, {})
M.Hide()
```

```
; Call TV_ItemSelect whenever a new item is
selected:
```

```
TV.OnEvent("ItemSelect",
Func("TV_ItemSelect").bind(DirList, SB, LV))
```

```
; Call Gui_Size whenever the window is resized:
```

```
Gui.OnEvent("Size", Func("Gui_Size").bind(TV, LV))
```

```
; Set the ListView's column widths (this is
optional):
```

```
Col2Width := 70 ; Narrow to reveal only the
YYYYMMDD part.
```

```
LV.ModifyCol(1, ListViewWidth - Col2Width - 30) ;
```

```
Allows room for vertical scrollbar.
```

```
LV.ModifyCol(2, Col2Width)
```

```
; Display the window. The OS will notify the
script whenever the user performs an eligible
action:
```

```

Gui.Show()

AddSubFoldersToTree(TV, Folder, DirList,
ParentItemID := 0)
{
 ; This function adds to the TreeView all
subfolders in the specified folder
 ; and saves their paths associated with an ID
into an object for later use.
 ; It also calls itself recursively to gather
nested folders to any depth.
 LoopFiles("%Folder%*.*", "D") ; Retrieve all
of Folder's sub-folders.
 {
 ItemID := TV.Add(A_LoopFileName, ParentItemID,
"Icon4")
 DirList[ItemID] := A_LoopFilePath
 DirList := AddSubFoldersToTree(TV,
A_LoopFilePath, DirList, ItemID)
 }
 return DirList
}

TV_ItemSelect(DirList, SB, LV, TV, Item) ; This
function is called when a new item is selected.
{
 ; Put the files into the ListView:
 LV.Delete() ; Clear all rows.
 LV.Opt("-Redraw") ; Improve performance by
disabling redrawing during load.
 TotalSize := 0 ; Init prior to loop below.
 LoopFiles(DirList[Item] "*.*") ; For
simplicity, omit folders so that only files are
shown in the ListView.
 {
 LV.Add(, A_LoopFileName,
A_LoopFileTimeModified)
 }
}

```

```
TotalSize += A_LoopFileSize
}
LV.Opt("+Redraw")
```

**; Update the three parts of the status bar to show info about the currently selected folder:**

```
SB.SetText(LV.GetCount() " files", 1)
SB.SetText(Round(TotalSize / 1024, 1) " KB", 2)
SB.SetText(DirList[Item], 3)
}
```

**Gui\_Size(TV, LV, Gui, MinMax, Width, Height) ; Expand/Shrink ListView and TreeView in response to the user's resizing.**

```
{
 if MinMax = -1 ; The window has been minimized.
 No action needed.
 return
 ; Otherwise, the window has been resized or
 maximized. Resize the controls to match.
 TV.Move("H" Height - 30) ; -30 for StatusBar
 and margins.
 LV.Move("H" Height - 30 " W" Width - TV.Pos.W -
 30)
}
```

# InputDialog

Displays an input box to ask the user to enter a string.

```
OutputVar := InputDialog([Text, Title, Options, Default])
```

```
Function Example: var := InputBox("Enter
Value")
```

## Parameters

### OutputVar

The name of the variable in which to store the text entered by the user.

### Text

The text of the input box, which is usually a message to the user indicating what kind of input is expected. If *Text* is long, it can be broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

### Title

The title of the input box. If omitted, it defaults to the current value of [A\\_ScriptName](#).

### Options

A string of case-insensitive options, with each separated from the last by a

space or tab.

**Xn Yn:** The X and Y coordinates of the dialog. For example, *X0 Y0* puts the window at the upper left corner of the desktop. If either coordinate is omitted, the dialog will be centered in that dimension. Either coordinate can be negative to position the dialog partially or entirely off the desktop (or on a secondary monitor in a multi-monitor setup).

**Wn Hn:** The width and height of the dialog. For example, *W375 H189* is the default size.

**T:** Specifies the timeout in seconds. For example, *T10.0* is ten seconds. If this value exceeds 2147483 (24.8 days), it will be set to 2147483. After the timeout has elapsed, the `InputBox` window will be automatically closed and `ErrorLevel` will be set to 2. `OutputVar` will still be set to what the user entered.

**Password:** Mask the user's input. To specify which character is used, follow this example: *Password\**

### Default

A string that will appear in the `InputBox`'s edit field when the dialog first appears. The user can change it by backspacing or other means.

### ErrorLevel

See below.

## Remarks

The dialog allows the user to enter text and then press OK or CANCEL. The user can resize the dialog window by dragging its borders.

`ErrorLevel` is set to 1 if the user presses the CANCEL button, 0 if the user presses OK, or 2 if the dialog times out. In all three cases, `OutputVar` is set to the value entered. This allows the CANCEL button to perform a function other than CANCEL should the script designer wish it.

A GUI window may display a modal `InputBox` by means of `OwnDialogs` option. A modal `InputBox` prevents the user from interacting with the GUI window until the `InputBox` is dismissed.

## Related

[GuiCreate](#), [Input](#), [MsgBox](#), [FileSelect](#), [DirSelect](#), [ToolTip](#)

## Example

```
InputBox, password, (your input will be hidden),
Enter Password, password
InputBox, UserInput, Please enter a phone number.,
Phone Number, w640 h480
if ErrorLevel
 MsgBox, CANCEL was pressed.
else
 MsgBox, You entered "%UserInput%"
```

# Menu

Creates, deletes, modifies and displays menus and menu items. Changes the tray icon and its tooltip. Controls whether the main window of a [compiled script](#) can be opened.

**Menu** MenuName, Cmd [, P3, P4, P5]

**Command Example:** Menu "MyMenu", "Add", "File", "MenuFile"

**Function Example:** Menu("MyMenu", "Add", "File", "MenuFile")

## Parameters

### MenuName

It can be TRAY or the name of any custom menu. A custom menu is automatically created the first time its name is used with the *Add* command. For example: `Menu, MyMenu, Add, Item1`.

Once created, a custom menu can be displayed with the *Show* command. It can also be attached as a submenu to one or more other menus via the *Add* command.

### Cmd, P3, P4, P5

These 4 parameters are dependent on each other. See list below for the allowed combinations.

## MenuItemName

The name or position of a menu item. Some common rules apply to this parameter across all sub-commands which use it:

To underline one of the letters in a menu item's name, precede that letter with an ampersand (&). When the menu is displayed, such an item can be selected by pressing the corresponding key on the keyboard. To display a literal ampersand, specify two consecutive ampersands as in this example: `Save && Exit`

When referring to an existing menu or menu item, the name is not case sensitive but any ampersands must be included. For example: `&Open`

To identify an existing item by its position in the menu, write the item's position followed by an ampersand. For example, `1&` indicates the first item.

## Add or Change Items in a Menu

**Add [, MenuItemName, Label-or-Submenu, Options]:** This is a multipurpose command that adds a menu item, updates one with a new submenu or label, or converts one from a normal item into a submenu (or vice versa). If *MenuItemName* does not yet exist, it will be added to the menu. Otherwise, *MenuItemName* is updated with the newly specified *Label-or-Submenu*.

To add a menu separator line, omit all three parameters.

The label subroutine is run as a new [thread](#) when the user selects the menu item (similar to [Gosub](#) and [hotkey subroutines](#)). If *Label-or-Submenu* is omitted, *MenuItemName* will be used as both the label and the menu item's name.

If it is not the name of an existing label, *Label-or-Submenu* can be the name of a function, or a single variable reference containing a [function object](#). For example, `%funcobj%` or `% funcobj`. Other expressions which return objects are currently unsupported. The function can optionally define parameters as shown below:

```
FunctionName(ItemName, ItemPos, MenuName)
```

To have *MenuItemName* become a submenu -- which is a menu item that opens a new menu when selected -- specify for *Label-or-Submenu* a colon followed by the *MenuName* of an existing custom menu. For example:

```
Menu, MySubmenu, add, Item1
```

```
Menu, tray, add, This Menu Item Is A Submenu,
:MySubmenu
```

If not omitted, *Options* must be a space- or tab-delimited list of one or more of the following options:

|                        |                                                                                                                                                                                                                                                                                                                                                 |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Pn</i>              | Replace <i>n</i> with the menu item's <a href="#">thread priority</a> , e.g. <code>P1</code> . If this option is omitted when adding a menu item, the priority will be 0, which is the standard default. If omitted when updating a menu item, the item's priority will not be changed. Use a decimal (not hexadecimal) number as the priority. |
| <code>+Radio</code>    | If the item is checked, a bullet point is used instead of a check mark.                                                                                                                                                                                                                                                                         |
| <code>+Right</code>    | The item is right-justified within the menu bar. This only applies to <a href="#">menu bars</a> , not popup menus or submenus.                                                                                                                                                                                                                  |
| <code>+Break</code>    | The item begins a new column in a popup menu.                                                                                                                                                                                                                                                                                                   |
| <code>+BarBreak</code> | As above, but with a dividing line between columns.                                                                                                                                                                                                                                                                                             |

The plus sign (+) is optional and can be replaced with minus (-) to remove the option, as in `-Radio`. Options are not case sensitive.

To change an existing item's options without affecting its label or submenu, simply omit the *Label-or-Submenu* parameter.

**Insert [*ItemToInsertBefore*, *NewItemName*, *Label-or-Submenu*, *Options*]:**

Inserts a new item before the specified item. Usage is identical to *Add* (above), except for the additional *ItemToInsertBefore* parameter, which is the name of an existing item or a position between 1 and [the current number of items](#) plus 1. Items can also be appended by omitting *ItemToInsertBefore* (by writing two

consecutive commas). Unlike *Add*, *Insert* creates a new item even if *NewItemName* matches the name of an existing item.

**Delete [, MenuItemName]:** Deletes *MenuItemName* from the menu. Standard menu items such as Exit (see below) cannot be individually deleted. If the *default* menu item is deleted, the effect will be similar to having used the *NoDefault* option. If *MenuItemName* is omitted, the entire *MenuName* menu will be deleted as will any menu items in other menus that use *MenuName* as a submenu. Deleting a menu also causes the current [Win32 menu](#) of its parent and submenus to be destroyed, to be recreated later as needed.

**DeleteAll:** Deletes all custom menu items from the menu, leaving the menu empty unless it contains the *standard* items (see below). Unlike a menu entirely deleted by the *Delete* command (see above), an empty menu still exists and thus any other menus that use it as a submenu will retain those submenus. However, the current [Win32 menus](#) of this menu and its parent and submenus are destroyed, to be recreated later as needed.

**Rename, MenuItemName [, NewName]:** Renames *MenuItemName* to *NewName* (if *NewName* is blank, *MenuItemName* will be converted into a separator line). The menu item's current target label or submenu is unchanged. A separator line can be converted to a normal item by specifying the position& of the separator and a non-blank *NewName*, and then using the *Add* command to give the item a label or submenu.

**Check, MenuItemName:** Adds a visible checkmark in the menu next to *MenuItemName* (if there isn't one already).

**Uncheck, MenuItemName:** Removes the checkmark (if there is one) from *MenuItemName*.

**ToggleCheck, MenuItemName:** Adds a checkmark if there wasn't one; otherwise, removes it.

**Enable, MenuItemName:** Allows the user to once again select *MenuItemName* if was previously disabled (grayed).

**Disable, MenuItemName:** Changes *MenuItemName* to a gray color to indicate that the user cannot select it.

**ToggleEnable, MenuItemName:** Disables *MenuItemName* if it was previously enabled; otherwise, enables it.

**Default [, MenuItemName]:** Changes the menu's default item to be *MenuItemName* and makes that item's font bold (setting a default item in menus other than TRAY is currently purely cosmetic). When the user double-clicks the tray icon, its default menu item is launched. If there is no default, double-clicking has no effect. If *MenuItemName* is omitted, the effect is the same as having used *NoDefault* below.

**NoDefault:** For the tray menu: Changes the menu back to having its standard default menu item, which is OPEN for non-compiled scripts and none for [compiled scripts](#) (except when the *MainWindow* option is in effect). If the OPEN menu item does not exist due to a previous use of the *NoStandard* command below, there will be no default and thus double-clicking the tray icon will have no effect. For menus other than TRAY: Any existing default item is returned to a

non-bold font.

**Standard:** Inserts the standard menu items at the bottom of the menu (if they are not already present). This command can be used with the tray menu or any other menu.

**NoStandard:** Removes all standard (non-custom) menu items from the tray menu (if they are present).

## Set or Remove a Menu Item's Icon

**Icon, MenuItemName, FileName [, IconNumber, IconWidth]:** Sets *MenuItemName*'s icon. *FileName* can either be an icon file or any image in a format supported by AutoHotkey. To use an icon group other than the first one in the file, specify its number for *IconNumber* (if omitted, it defaults to 1). If *IconNumber* is negative, its absolute value is assumed to be the resource ID of an icon within an executable file. Specify the desired width of the icon in *IconWidth*. If the icon group indicated by *IconNumber* contains multiple icon sizes, the closest match is used and the icon is scaled to the specified size. See the Examples section for usage examples.

Currently it is necessary to specify "actual size" when setting the icon to preserve transparency on Windows Vista and later. For example:

```
Menu, MenuName, Icon, MenuItemName,
Filename.png,, 0
```

Known limitation: Icons on Gui menu bars are positioned incorrectly on Windows XP and older.

A [bitmap or icon handle](#) can be used instead of a filename. For example, `HBITMAP:%handle%`.

**NoIcon, MenuItemName:** Removes *MenuItemName*'s icon, if any.

## Change the Tray Icon or ToolTip (MenuName must be TRAY)

**Icon [, FileName, IconNumber, 1]:** Changes the script's icon to one of the ones from *FileName*. The following types of files are supported: ICO, CUR, ANI, EXE, DLL, CPL, SCR, and other types that contain icon resources. To use an icon group other than the first one in the file, specify its number for *IconNumber* (if omitted, it defaults to 1). For example, 2 would load the default icon from the second icon group. If *IconNumber* is negative, its absolute value is assumed to be the resource ID of an icon within an executable file. Specify an asterisk (\*) for *FileName* to restore the script to its default icon.

The last parameter: Specify 1 for the last parameter to freeze the icon, or 0 to unfreeze it (or leave it blank to keep the frozen/unfrozen state unchanged). When the icon has been frozen, [Pause](#) and [Suspend](#) will not change it. Note: To freeze or unfreeze the *current* icon, use 1 or 0 as in the following example: `Menu, Tray, Icon,,, 1`.

Changing the tray icon also changes the icon displayed by [InputBox](#) and subsequently-created [GUI](#) windows. [Compiled scripts](#) are also affected even if a custom icon was specified at the time of compiling. Note: Changing the icon will not unhide the tray icon if it was previously hidden by means such as [#NoTrayIcon](#); to do that, use `Menu, Tray, Icon` (with no parameters).

Slight distortion may occur when loading tray icons from file types other than .ICO. This is especially true for 16x16 icons. To prevent this, store the desired

tray icon inside a .ICO file.

There are some icons built into the operating system's DLLs and CPLs that might be useful. For example: `Menu, Tray, Icon, Shell32.dll, 174`.

The built-in variables `A_IconNumber` and `A_IconFile` contain the number and name (with full path) of the current icon (both are blank if the icon is the default).

A [bitmap or icon handle](#) can be used instead of a filename. For example, `HBITMAP:%handle%`.

**Icon** (with no parameters): Creates the tray icon if it isn't already present. This will override `#NoTrayIcon` if that directive is also present in the script.

**NoIcon**: Removes the tray icon if it exists. If this command is used at the very top of the script, the tray icon might be briefly visible when the script is launched. To prevent that, use `#NoTrayIcon` instead. The built-in variable `A_IconHidden` contains 1 if the tray icon is currently hidden or 0 otherwise.

**Tip [, Text]**: Changes the tray icon's tooltip -- which is displayed when the mouse hovers over it. To create a multi-line tooltip, use the linefeed character (`\n`) in between each line, e.g. `Line1\nLine2`. Only the first 127 characters of `Text` are displayed, and `Text` is truncated at the first tab character, if present. If `Text` is omitted, the tooltip is restored to its default text. The built-in variable `A_IconTip` contains the current text of the tooltip (blank if the text is at its default).

## Miscellaneous Commands

**Show [, X, Y]:** Displays *MenuName*, allowing the user to select an item with arrow keys, menu shortcuts (underlined letters), or the mouse. Any menu can be shown, including the tray menu but with the exception of [GUI menu bars](#). If both X and Y are omitted, the menu is displayed at the current position of the mouse cursor. If only one of them is omitted, the mouse cursor's position will be used for it. X and Y are relative to the active window. Specify "[CoordMode](#), Menu" beforehand to make them relative to the entire screen.

**Color, ColorValue [, Single]:** Changes the background color of the menu to *ColorValue*, which is one of the 16 primary HTML color names or a 6-digit RGB color value (see [color chart](#)). Leave *ColorValue* blank (or specify the word Default) to restore the menu to its default color. If the word Single is not present as the next parameter, any submenus attached to this menu will also be changed in color.

**Click, ClickCount:** Specify 1 for *ClickCount* to allow a single-click to activate the tray menu's default menu item. Specify 2 for *ClickCount* to return to the default behavior (double-click). For example: `Menu, Tray, Click, 1`.

**MainWindow:** This command affects [compiled scripts](#) only. It allows the script's main window to be opened via the tray icon, which is otherwise impossible. It also enables the items in the main window's View menu such as "Lines most recently executed", which allows viewing of the script's source code and other info. *MenuName* must be TRAY.

**NoMainWindow** (default): This command affects [compiled scripts](#) only. It restores the script to its default behavior, that is, it prevents the main window from being opened. Even while this option is in effect, the following commands are still able to show the main window when they are encountered in the script at runtime: [ListLines](#), [ListVars](#), [ListHotkeys](#), and [KeyHistory](#). *MenuName* must be TRAY.

**UseErrorLevel [, off]**: If this option is never used in the script, it defaults to OFF. The OFF setting displays a dialog and terminates the [current thread](#) whenever the Menu command generates an error. Specify [Menu, Tray, UseErrorLevel](#) to prevent the dialog and thread termination; instead, [ErrorLevel](#) will be set to 1 if there was a problem or 0 otherwise. To turn this option back off, specify OFF for the next parameter. This setting is global, meaning it affects all menus, not just *MenuName*.

## Win32 Menu

As items are added to a menu or modified, the name and other properties of each item are recorded by the Menu command, but the actual [Win32 menu](#) is not constructed immediately. This occurs when the menu or its parent menu is attached to a GUI or shown, either for the first time or if the menu has been "destroyed" since it was last shown. Any of the following can cause this Win32 menu to be destroyed, along with any parent menus and submenus:

- Deleting a menu.
- Replacing an item's submenu with a label or a different menu.
- Menu, *MenuName*, **DeleteAll**.
- Menu, *MenuName*, **NoStandard** (if the standard items were present).

Any modifications which are made to the menu directly by Win32 API calls only apply to the current "instance" of the menu, and are lost when the menu is destroyed.

Each menu item is assigned an ID when it is first added to the menu. Scripts cannot rely on an item receiving a particular ID, but can retrieve the ID of an item by using `GetMenuItemID` as shown in the [MenuGetHandle example](#). This ID cannot be used with the Menu command, but can be used with various [Win32 functions](#).

## Remarks

The names of menus and menu items can be up to 260 characters long.

Separator lines can be added to the menu by using `Menu, MenuName, Add` (i.e. omit all other parameters). To delete separator lines individually, identify them by their position in the menu (requires v1.1.23+). For example, use `Menu, MenuName, Delete, 3&` if there are two items preceding the separator. Alternatively, use `Menu, MenuName, DeleteAll` and then re-add your custom menu items.

New menu items are always added at the bottom of the menu. For the tray menu: To put your menu items on top of the standard menu items (after adding your own menu items) run `Menu, Tray, NoStandard` followed by `Menu, Tray, Standard`.

The standard menu items such as "Pause Script" and "Suspend Hotkeys" cannot be individually operated upon by any menu sub-command.

If a menu ever becomes completely empty -- such as by using `Menu, MyMenu, DeleteAll` -- it cannot be shown. If the tray menu becomes empty, right-clicking and double-clicking the tray icon will have no effect (in such cases it is usually better to use `#NoTrayIcon`).

If a menu item's subroutine is already running and the user selects the same menu item again, a new [thread](#) will be created to run that same subroutine, interrupting the previous thread. To instead buffer such events until later, use

**Critical** as the subroutine's first line (however, this will also buffer/defer other threads such as the press of a hotkey).

Whenever a subroutine is launched via a menu item, it starts off fresh with the default values for settings such as **SendMode**. These defaults can be changed in the **auto-execute section**.

The built-in variables **A\_ThisMenuItem** and **A\_ThisMenuItemPos** contain the name and position of the custom menu item most recently selected by the user (blank if none). Similarly, **A\_ThisMenu** is the name of the menu from which **A\_ThisMenuItem** was selected. These variables are useful when building a menu whose contents are not always the same. In such a case, it is usually best to point all such menu items to the same label and have that label refer to the above variables to determine what action to take.

## Related

[GUI](#), [Threads](#), [Thread](#), [Critical](#), [#NoTrayIcon](#), [Gosub](#), [Return](#), [SetTimer](#)

## Examples

```
; EXAMPLE #1: This is a working script that adds a
new menu item to the bottom of the tray icon menu.
```

```
Menu, tray, add ; Creates a separator line.
Menu, tray, add, Item1, MenuHandler ; Creates a
new menu item.
return
```

```
MenuHandler:
MsgBox You selected %A_ThisMenuItem% from menu
%A_ThisMenu%.
return
```

```
; EXAMPLE #2: This is a working script that
creates a popup menu that is displayed when the
user presses the Win-Z hotkey.
```

```
; Create the popup menu by adding some items to
it.
```

```
Menu, MyMenu, Add, Item1, MenuHandler
Menu, MyMenu, Add, Item2, MenuHandler
Menu, MyMenu, Add ; Add a separator line.
```

```
; Create another menu destined to become a submenu
of the above menu.
```

```
Menu, Submenu1, Add, Item1, MenuHandler
Menu, Submenu1, Add, Item2, MenuHandler
```

```
; Create a submenu in the first menu (a right-
arrow indicator). When the user selects it, the
```

second menu is displayed.

```
Menu, MyMenu, Add, My Submenu, :Submenu1
```

```
Menu, MyMenu, Add ; Add a separator line below
the submenu.
```

```
Menu, MyMenu, Add, Item3, MenuHandler ; Add
another menu item beneath the submenu.
```

```
return ; End of script's auto-execute section.
```

```
MenuHandler:
```

```
MsgBox You selected %A_ThisMenuItem% from the menu
%A_ThisMenu%.
```

```
return
```

```
#z::Menu, MyMenu, Show ; i.e. press the Win-Z
hotkey to show the menu.
```

; EXAMPLE #3: This is a working script that demonstrates some of the various menu commands.

```
#SingleInstance
```

```
menu, tray, add ; separator
```

```
menu, tray, add, TestToggle&Check
```

```
menu, tray, add, TestToggleEnable
```

```
menu, tray, add, TestDefault
```

```
menu, tray, add, TestStandard
```

```
menu, tray, add, TestDelete
```

```
menu, tray, add, TestDeleteAll
```

```
menu, tray, add, TestRename
```

```
menu, tray, add, Test
```

```
return
```

```
////////////////////////////////////
```

```
TestToggle&Check:
menu, tray, ToggleCheck, TestToggle&Check
menu, tray, Enable, TestToggleEnable ; Also
enables the next test since it can't undo the
disabling of itself.
menu, tray, add, TestDelete ; Similar to above.
return
```

```
TestToggleEnable:
menu, tray, ToggleEnable, TestToggleEnable
return
```

```
TestDefault:
if default = "TestDefault"
{
 menu, tray, NoDefault
 default := ""
}
else
{
 menu, tray, Default, TestDefault
 default := "TestDefault"
}
return
```

```
TestStandard:
if standard <> "n"
{
 menu, tray, NoStandard
 standard := "n"
}
else
{
 menu, tray, Standard
 standard := "y"
}
return
```

```
TestDelete:
menu, tray, delete, TestDelete
return

TestDeleteAll:
menu, tray, DeleteAll
return

TestRename:
if NewName <> "renamed"
{
 OldName := "TestRename"
 NewName := "renamed"
}
else
{
 OldName := "renamed"
 NewName := "TestRename"
}
menu, tray, rename, %OldName%, %NewName%
return

Test:
MsgBox, You selected "%A_ThisMenuItem%" in menu
"%A_ThisMenu%".
return
```

**; EXAMPLE #4: This is a working script that adds icons to its menu items.**

```
Menu("FileMenu", "Add", "Script Icon",
"MenuHandler")
Menu("FileMenu", "Add", "Suspend Icon",
"MenuHandler")
Menu("FileMenu", "Add", "Pause Icon",
"MenuHandler")
```

```
Menu("FileMenu", "Icon", "Script Icon", A_AhkPath,
2) ; 2nd icon group from the file
Menu("FileMenu", "Icon", "Suspend Icon",
A_AhkPath, -206) ; icon with resource ID 206
Menu("FileMenu", "Icon", "Pause Icon", A_AhkPath,
-207) ; icon with resource ID 207
Menu("MyMenuBar", "Add", "&File", ":FileMenu")
Gui := GuiCreate()
Gui.Menu := "MyMenuBar"
Gui.Add("Button",, "Exit This
Example").OnEvent("Click", "Exit_Click")
Gui.Show()

MenuHandler() {
}

Exit_Click() {
 WinClose()
}
```

# MsgBox

Displays the specified text in a small window containing one or more buttons (such as Yes and No).

```
MsgBox [Text, Title, Options]
Result := MsgBox([Text, Title, Options])
```

## Parameters

### Text

If all the parameters are omitted, the MsgBox will display the text "Press OK to continue.". Otherwise, this parameter is the text displayed inside the message box to instruct the user what to do, or to present information.

[Escape sequences](#) can be used to denote special characters. For example, `\n` indicates a linefeed character, which ends the current line and begins a new one. Thus, using `text1\n\ntext2` would create a blank line between `text1` and `text2`.

If *Text* is long, it can be broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

### Title

The title of the message box window. If omitted, it defaults to the current

value of `A_ScriptName`.

## Options

Indicates the type of message box and the possible button combinations. If blank or omitted, it defaults to 0. See the table below for allowed values.

The *Options* parameter accepts either an integer value, which is passed directly to the operating system's `MessageBox` function, or a string of zero or more case-insensitive options separated by at least one space or tab. One or more numeric options may also be included in the string.

| Function                                     | Decimal Value | Hex Value | String Value                                                                 |
|----------------------------------------------|---------------|-----------|------------------------------------------------------------------------------|
| OK (that is, only an OK button is displayed) | 0             | 0x0       | <code>OK</code> or <code>0</code>                                            |
| OK/Cancel                                    | 1             | 0x1       | <code>OKCancel</code> , <code>O/C</code> or <code>OC</code>                  |
| Abort/Retry/Ignore                           | 2             | 0x2       | <code>AbortRetryIgnore</code> , <code>A/R/I</code> or <code>ARI</code>       |
| Yes/No/Cancel                                | 3             | 0x3       | <code>YesNoCancel</code> , <code>Y/N/C</code> or <code>YNC</code>            |
| Yes/No                                       | 4             | 0x4       | <code>YesNo</code> , <code>Y/N</code> or <code>YN</code>                     |
| Retry/Cancel                                 | 5             | 0x5       | <code>RetryCancel</code> , <code>R/C</code> or <code>RC</code>               |
| Cancel/Try Again/Continue                    | 6             | 0x6       | <code>CancelTryAgainContinue</code> , <code>C/T/C</code> or <code>CTC</code> |
| Adds a Help button (see remarks below)       | 16384         | 0x4000    |                                                                              |
| Icons                                        |               |           |                                                                              |

|                                                                                  |         |          |          |
|----------------------------------------------------------------------------------|---------|----------|----------|
| Icon Hand (stop/error)                                                           | 16      | 0x10     | Iconx    |
| Icon Question                                                                    | 32      | 0x20     | Icon?    |
| Icon Exclamation                                                                 | 48      | 0x30     | Icon!    |
| Icon Asterisk (info)                                                             | 64      | 0x40     | Iconi    |
| Default Button                                                                   |         |          |          |
| Makes the 2nd button the default                                                 | 256     | 0x100    | Default2 |
| Makes the 3rd button the default                                                 | 512     | 0x200    | Default3 |
| Makes the 4th button the default (requires the Help button to be present)        | 768     | 0x300    | Default4 |
| Mode                                                                             |         |          |          |
| System Modal (always on top)                                                     | 4096    | 0x1000   |          |
| Task Modal                                                                       | 8192    | 0x2000   |          |
| Always-on-top (style WS_EX_TOPMOST) (like System Modal but omits title bar icon) | 262144  | 0x40000  |          |
| Make the text right-justified                                                    | 524288  | 0x80000  |          |
| Right-to-left reading order for Hebrew/Arabic                                    | 1048576 | 0x100000 |          |
| Additional string-only options                                                   |         |          |          |
|                                                                                  |         |          |          |

**Owner:** To specify an [owner window](#) for the MsgBox, use the word `Owner` followed immediately by a HWND (window ID).

`Owner%HWND%`

**Timeout:** To have the MsgBox close automatically if the user has not closed it within a specified time, use the letter `T` followed by the timeout in seconds, which can contain a decimal point. If this value exceeds 2147483 (24.8 days), it will be set to 2147483.

If the MsgBox times out, the [return value](#) is the word `Timeout`.

Known limitation: If the MsgBox contains only an OK button, the [return value](#) will indicate that the OK button was pressed if the MsgBox times out while its own [thread](#) is interrupted by another.

## Return Value

When called from an expression, `MsgBox()` returns one of the following strings to represent which button the user pressed:

OK

Cancel

Yes

No

Abort

Retry

Ignore

TryAgain

Continue

Timeout (that is, the word "timeout" is returned if the MsgBox timed out)

## Remarks

When using the command syntax, remember to [escape](#) any commas in the text or title which are intended to be literal.

To determine which button the user pressed, use the function's [return value](#). For example:

```
Result := MsgBox("Would you like to continue?
(press Yes or No)",, "YesNo")
if Result = "Yes"
 MsgBox You pressed Yes.
else
 MsgBox You pressed No.

if MsgBox("Retry or cancel?",, "R/C") = "Retry"
 MsgBox("You pressed Retry.")
```

The names of the buttons can be customized by following [this example](#).

**Tip:** Pressing Control-C while a MsgBox window is active will copy its text to the clipboard. This applies to all MsgBoxes, not just those produced by AutoHotkey.

**Using MsgBox with GUI windows:** A GUI window may display a *modal* MsgBox by means of the [OwnDialogs option](#). A *modal* MsgBox prevents the user from interacting with the GUI window until the MsgBox is dismissed. In such a case, it is not necessary to specify the System Modal or Task Modal options from the table above.

When the [OwnDialogs option](#) is *not* in effect, the Task Modal option (8192) can be used to disable all the script's windows until the user dismisses the MsgBox.

If the [Owner%HWND%](#) option is specified, it takes precedence over any other setting. *HWND* can be the HWND of any window, even one not owned by the script.

**The Help button:** When the Help button option (83) is present in *Options*, pressing the Help button will have no effect unless both of the following are true:

1. The MsgBox is owned by a GUI window by means of the [OwnDialogs option](#).
2. The script is monitoring the WM\_HELP message (0x53). For example: [OnMessage\(0x53, "WM\\_HELP"\)](#). When the WM\_HELP() function is called, it may guide the user by means such as showing another window or MsgBox.

**The Close button (in MsgBox's title bar):** Since the MsgBox window is a built-in feature of the operating system, its X button is enabled only when certain buttons are present. If there is only an OK button, clicking the X button is the

same as pressing OK. Otherwise, the X button is disabled unless there is a Cancel button, in which case clicking the X is the same as pressing Cancel.

## Related

[InputDialog](#), [FileSelect](#), [DirSelect](#), [ToolTip](#), [GuiCreate](#)

## Example

### Function Syntax

```
MsgBox() ; "Press OK to continue."
```

```
MsgBox("Commas (,) do not need to be escaped in
quoted strings.")
```

```
MsgBox("This MsgBox has a custom title.", "A
Custom Title")
```

```
MsgBox(" ; Use a continuation section to span
multiple lines:
```

```
(
 The first parameter is displayed as the
message.
 The second parameter becomes the window title.
 The third parameter determines the type of
message box.
)", "Window Title", "iconi")
```

```
if MsgBox("Do you want to continue? (Press YES or
NO)",, "YesNo") = "No"
 return
```

```
result := MsgBox("This MsgBox will time out in 5
```

```
seconds. Continue?",, "Y/N T5")
if result = "Timeout"
 MsgBox("You didn't press YES or NO within the
5-second period.")
else if result = "No"
 return
```

**; Include a variable or sub-expression in the message.**

```
var := 10
MsgBox("The initial value is: " var)
MsgBox("The result is: " var * 2) ;
```

**Concatenation.**

```
MsgBox("The result is: %var*2%") ; Include an
expression in a string with %%.
```

## Command Syntax

```
MsgBox ; "Press OK to continue."
```

MsgBox Commas (`,) in the text must be escaped when using this syntax.

```
MsgBox This MsgBox has a custom title., A Custom Title
```

MsgBox, ; Use a continuation section to span multiple lines:

```
(
 The first parameter is displayed as the message.
 The second parameter becomes the window title.
 The third parameter determines the type of message box.
), Window Title, iconi
```

```
MsgBox, Do you want to continue? (Press YES or
NO),, YesNo, result
if result = "No"
 return
```

```
MsgBox, This MsgBox will time out in 5 seconds.
Continue?,, Y/N T5, result
if result = "Timeout"
 MsgBox, You didn't press YES or NO within the
5-second period.
else if result = "No"
 return
```

**;Include a variable in the message.**

```
var := 10
```

```
MsgBox The initial value is: %var% ; Enclose
```

**variable names in percent signs.**

```
MsgBox % "The result is: " var * 2 ; The "% "
```

**prefix makes this whole parameter an expression.**

```
MsgBox The result is: %var*2% ; Expressions are
also allowed within %%.
```

# OnMessage()

Specifies a [function](#) or [function object](#) to call automatically when the script receives the specified message.

```
OutputVar := OnMessage(MsgNumber, WindowId,
Function, MaxThreads)
```

```
Command Example: OnMessage 0x200, "MyFunc"
Function Example: Func := OnMessage(0x200,
"MyFunc")
```

## Parameters

### OutputVar

The name of the variable in which to store the function name if a function was already monitoring *MsgNumber*. Otherwise, the return value is blank even if a new function was put into effect.

### MsgNumber

The number of the message to monitor or query, which should be between 0 and 4294967295 (0xFFFFFFFF). If you do not wish to monitor a [system message](#) (that is, one below 0x400), it is best to choose a number greater than 4096 (0x1000) to the extent you have a choice. This reduces the chance of interfering with messages used internally by current and future versions of AutoHotkey.

## WindowId (optional)

The window whose messages will be monitored. This parameter can be omitted.

## Function

A [function's](#) name or a [function object](#). To pass a literal function name, it must be enclosed in quotes.

## MaxThreads

This integer is normally omitted, in which case the monitor function is limited to one [thread](#) at a time. This is usually best because otherwise, the script would process messages out of chronological order whenever the monitor function interrupts itself. Therefore, as an alternative to *MaxThreads*, consider using *Critical* as described [below](#).

By default, when multiple functions are registered for a single *MsgNumber*, they are called in the order that they were registered. To register a function to be called before any previously registered functions, specify a negative value for *MaxThreads*. For example,

```
OnMessage(Msg, Fn, -2)
```

registers `Fn` to be called before any other functions previously registered for *Msg*, and allows *Fn* a maximum of 2 threads. However, if the function is already registered, the order will not change unless it is unregistered and then re-registered.

## Usage

Any number of functions or [function objects](#) can monitor a given *MsgNumber*.

Either of these two lines registers a function object to be called **after** any previously registered functions:

```
OnMessage MsgNumber, Function ; Option 1 - omit
MaxThreads
OnMessage MsgNumber, Function, 1 ; Option 2 -
specify MaxThreads 1
```

This registers a function object to be called **before** any previously registered functions:

```
OnMessage MsgNumber, Function, -1
```

To unregister a function object, specify 0 for *MaxThreads*:

```
OnMessage MsgNumber, Function, 0
```

## Failure

An exception is thrown if *Function*:

1. is not an object or the name of a user-defined function; or
2. is known to require more than four parameters.

## The Function's Parameters

A [function](#) assigned to monitor one or more messages can accept up to four

parameters:

```
MyMessageMonitor(wParam, lParam, msg, hwnd)
{
 ... body of function...
}
```

Although the names you give the parameters do not matter, the following information is sequentially assigned to them:

Parameter #1: The message's WPARAM value.

Parameter #2: The message's LPARAM value.

Parameter #3: The message number, which is useful in cases where a function monitors more than one message.

Parameter #4: The HWND (unique ID) of the window or control to which the message was sent. The HWND can be used with [ahk\\_id](#).

WPARAM and LPARAM are unsigned 32-bit integers (from 0 to  $2^{32}-1$ ) or signed 64-bit integers (from  $-2^{63}$  to  $2^{63}-1$ ) depending on whether the exe running the script is 32-bit or 64-bit. For 32-bit scripts, if an incoming parameter is intended to be a signed integer, any negative numbers can be revealed by following this example:

```
if (A_PtrSize = 4 && wParam > 0x7FFFFFFF) ;
 Checking A_PtrSize ensures the script is 32-
 bit.
 wParam := -(-wParam) - 1
```

You can omit one or more parameters from the end of the list if the

corresponding information is not needed. For example, a function defined as `MyMsgMonitor(wParam, lParam)` would receive only the first two parameters, and one defined as `MyMsgMonitor()` would receive none of them.

## Additional Information Available to the Function

In addition to the parameters received above, the function may also consult the built-in variable `A_EventInfo`, which contains 0 if the message was sent via `SendMessage`. If sent via `PostMessage`, it contains the [tick-count time](#) the message was posted.

A monitor function's [last found window](#) starts off as the parent window to which the message was sent (even if it was sent to a control). If the window is hidden but not a GUI window (such as the script's main window), turn on `DetectHiddenWindows` before using it. For example:

```
DetectHiddenWindows On
MsgParentWindow := WinExist() ; This stores
the unique ID of the window to which the
message was sent.
```

## What the Function Should Return

If a monitor function uses `Return` without any parameters, or it specifies a blank value such as `""` (or it never uses `Return` at all), the incoming message goes on to be processed normally when the function finishes. The same thing happens if the function `Exits` or causes a runtime error such as [running](#) a nonexistent file. By

contrast, returning an integer causes it to be sent immediately as a reply; that is, the program does not process the message any further. For example, a function monitoring `WM_LBUTTONDOWN` (0x201) may return an integer to prevent the target window from being notified that a mouse click has occurred. In many cases (such as a message arriving via `PostMessage`), it does not matter which integer is returned; but if in doubt, 0 is usually safest.

The range of valid return values depends on whether the exe running the script is 32-bit or 64-bit. Non-empty return values must be between  $-2^{31}$  and  $2^{32}-1$  for 32-bit scripts (`A_PtrSize = 4`) and between  $-2^{63}$  and  $2^{63}-1$  for 64-bit scripts (`A_PtrSize = 8`).

If there are multiple functions monitoring a given message number, they are called one by one until one returns a non-empty value.

## General Remarks

Unlike a normal function-call, the arrival of a monitored message calls the function as a new `thread`. Because of this, the function starts off fresh with the default values for settings such as `SendMode` and `DetectHiddenWindows`. These defaults can be changed in the `auto-execute` section.

Messages sent to a control (rather than being posted) are not monitored because the system routes them directly to the control behind the scenes. This is seldom an issue for system-generated messages because most of them are posted.

Any script with active message monitors is automatically `persistent`, which

means that it will not exit until `ExitApp` is used.

If a message arrives while its function is still running due to a previous arrival of the same message, the function will not be called again (except if `MaxThreads` is greater than 1); instead, the message will be treated as unmonitored. If this is undesirable, a message greater than or equal to 0x312 can be buffered until its function completes by specifying `Critical` as the first line of the function.

Alternatively, `Thread Interrupt` can achieve the same thing as long as it lasts long enough for the function to finish. By contrast, a message less than 0x312 cannot be buffered by `Critical` or `Thread Interrupt` (however, `Critical` may help because it checks messages *less often*, which gives the function more time to finish). The only way to guarantee that no such messages are missed is to ensure the function finishes in under 6 milliseconds (though this limit can be raised via `Critical 30`). One way to do this is to have it queue up a future thread by *posting* to its own script a monitored message number higher than 0x312. That message's function should use `Critical` as its first line to ensure that its messages are buffered.

If a monitored message that is numerically less than 0x312 arrives while the script is absolutely uninterruptible -- such as while a `menu` is displayed, a `KeyDelay/MouseDelay` is in progress, or the clipboard is being *opened* -- the message's function will not be called and the message will be treated as unmonitored. By contrast, a monitored message of 0x312 or higher will be buffered during these uninterruptible periods; that is, its function will be called when the script becomes interruptible.

If a monitored message numerically less than 0x312 arrives while the script is uninterruptible merely due to the settings of `Thread Interrupt` or `Critical`, the

current thread will be interrupted so that the function can be called. By contrast, a monitored message of 0x312 or higher will be buffered until the thread finishes or becomes interruptible.

The [priority](#) of OnMessage threads is always 0. Consequently, no messages are monitored or buffered when the current thread's priority is higher than 0.

Caution should be used when monitoring system messages (those below 0x400). For example, if a monitor function does not finish quickly, the response to the message might take longer than the system expects, which might cause side-effects. Unwanted behavior may also occur if a monitor function returns an integer to suppress further processing of a message, but the system expected different processing or a different response.

When the script is displaying a system dialog such as [MsgBox](#), any message posted to a control is not monitored. For example, if the script is displaying a MsgBox and the user clicks a button in a GUI window, the WM\_LBUTTONDOWN message is sent directly to the button without calling the monitor function.

Although an external program may post messages directly to a script's thread via PostThreadMessage() or other API call, this is not recommended because the messages would be lost if the script is displaying a system window such as a [MsgBox](#). Instead, it is usually best to post or send the messages to the script's main window or one of its GUI windows.

## Related

RegisterCallback, OnExit, OnClipboardChange, Post/SendMessage, Functions,  
List of Windows Messages, Threads, Critical, DllCall

## Examples

**; Example: The following is a working script that monitors mouse clicks in a GUI window.**

**; Related topic: [ContextMenu event](#)**

```
Gui := GuiCreate(, "Example Window")
Gui.Add("Text",, "Click anywhere in this window.")
Gui.Add("Edit", "w200")
Gui.OnEvent("Close", "Gui_Close")
Gui.Show()
OnMessage(0x201, "WM_LBUTTONDOWN")

WM_LBUTTONDOWN(wParam, lParam, msg, hwnd)
{
 X := lParam & 0xFFFF
 Y := lParam >> 16
 Gui := GuiFromHwnd(hwnd)
 GuiControl := GuiCtrlFromHwnd(hwnd)
 if GuiControl
 {
 Gui := GuiControl.Gui
 Control := "`n(in control " .
GuiControl.ClassNN . ")"
 }
 ToolTip("You left-clicked in Gui window
'%Gui.Title%' at client coordinates %X%X%Y%."
Control)
}

Gui_Close() {
 ExitApp()
}
```

```
}
```

```
; Example: The following script detects system shutdown/logoff and allows you to abort it (this is reported NOT to work on Windows Vista and later).
; Related topic: OnExit
```

```
; The following DllCall is optional: it tells the OS to shut down this script first (prior to all other applications).
```

```
DllCall("kernel32.dll\SetProcessShutdownParameters", UInt, 0x4FF, UInt, 0)
OnMessage(0x11, "WM_QUERYENDSESSION")
return
```

```
WM_QUERYENDSESSION(wParam, lParam)
```

```
{
```

```
 ENDESSION_LOGOFF := 0x80000000
```

```
 if (lParam & ENDESSION_LOGOFF) ; User is logging off.
```

```
 EventType := "Logoff"
```

```
 else ; System is either shutting down or restarting.
```

```
 EventType := "Shutdown"
```

```
 Result := MsgBox("%EventType% in progress.
```

```
Allow it?",, 4)
```

```
 ; Tell the OS to allow the shutdown/logoff to continue only if the user clicked Yes.
```

```
 return Result = "Yes"
```

```
}
```

```
; Example: Have a script receive a custom message and up to two numbers from some other script or
```

```
program
; (to send strings rather than numbers, see the
example after this one).
```

```
OnMessage(0x5555, "MsgMonitor")
OnMessage(0x5556, "MsgMonitor")
```

```
MsgMonitor(wParam, lParam, msg)
{
```

```
 ; Since returning quickly is often important,
it is better to use a ToolTip than
 ; something like MsgBox that would prevent the
function from finishing:
```

```
 ToolTip Message %msg% arrived: `nWPARAM:
%wParam% `nLPARAM: %lParam%
}
```

```
; The following could be used inside some other
script to run the function inside the above
script:
```

```
SetTitleMatchMode 2
DetectHiddenWindows On
if WinExist("Name of Receiving Script.ahk
ahk_class AutoHotkey")
 PostMessage, 0x5555, 11, 22 ; The message is
sent to the "last found window" due to WinExist()
above.
DetectHiddenWindows Off ; Must not be turned off
until after PostMessage.
```

```
; Example: Send a string of any length from one
script to another. This is a working example.
; To use it, save and run both of the following
scripts then press Win+Space to show an
; InputBox that will prompt you to type in a
string.
```

**; Save the following script as "Receiver.ahk" then launch it:**

```
#SingleInstance
OnMessage(0x4a, "Receive_WM_COPYDATA") ; 0x4a is
WM_COPYDATA
return

Receive_WM_COPYDATA(wParam, lParam)
{
 StringAddress := NumGet(lParam + 2*A_PtrSize)
; Retrieves the CopyDataStruct's lpData member.
 CopyOfData := StrGet(StringAddress) ; Copy
the string out of the structure.
; Show it with ToolTip vs. MsgBox so we can
return in a timely fashion:
 ToolTip %A_ScriptName%\nReceived the following
string:\n%CopyOfData%
 return true ; Returning 1 (true) is the
traditional way to acknowledge this message.
}
```

**; Save the following script as "Sender.ahk" then launch it. After that, press the Win+Space hotkey.**

```
TargetScriptTitle := "Receiver.ahk ahk_class
AutoHotkey"

#space:: ; Win+Space hotkey. Press it to show an
InputBox for entry of a message string.
InputBox, StringToSend, Enter some text to Send:,
Send text via WM_COPYDATA
if ErrorLevel ; User pressed the Cancel button.
 return
result := Send_WM_COPYDATA(StringToSend,
TargetScriptTitle)
if result = "ERROR"
```

```
 MsgBox SendMessage failed. Does the following
WinTitle exist?:`n%TargetScriptTitle%
else if result = 0
```

```
 MsgBox Message sent but the target window
responded with 0, which may mean it ignored it.
return
```

```
Send_WM_COPYDATA(ByRef StringToSend, ByRef
TargetScriptTitle) ; ByRef saves a little memory
in this case.
```

```
; This function sends the specified string to the
specified window and returns the reply.
```

```
; The reply is 1 if the target window processed
the message, or 0 if it ignored it.
```

```
{
```

```
 VarSetCapacity(CopyDataStruct, 3*A_PtrSize, 0)
; Set up the structure's memory area.
```

```
 ; First set the structure's cbData member to
the size of the string, including its zero
terminator:
```

```
 SizeInBytes := (StrLen(StringToSend) + 1) * 2
 NumPut(SizeInBytes, CopyDataStruct, A_PtrSize)
; OS requires that this be done.
```

```
 NumPut(&StringToSend, CopyDataStruct,
2*A_PtrSize) ; Set lpData to point to the string
itself.
```

```
 Prev_DetectHiddenWindows :=
A_DetectHiddenWindows
```

```
 Prev_TitleMatchMode := A_TitleMatchMode
 DetectHiddenWindows On
 SetTitleMatchMode 2
```

```
 SendMessage, 0x4a, 0, % &CopyDataStruct,,
%TargetScriptTitle% ; 0x4a is WM_COPYDATA. Must
use Send not Post.
```

```
 DetectHiddenWindows %Prev_DetectHiddenWindows%
; Restore original setting for the caller.
```

```
 SetTitleMatchMode %Prev_TitleMatchMode%
```

```
; Same.
 return ErrorLevel ; Return SendMessage's
reply back to our caller.
}
```

```
; Example: See the WinLIRC client script for a
demonstration of how to use OnMessage() to receive
; notification when data has arrived on a network
connection.
```

# Progress

Creates or updates a window containing a progress bar.

```
Progress ProgressParam1 [, SubText, MainText,
WinTitle, FontName]
Progress, Off
```

```
Command Example: Progress 50, "copy files...",
"Instaling`, please wait...", "Installing
MyProgress"
Function Example: Progress(50, "copy
files...", "Installing, please
wait...", "Installing MyProgress")
```

## Parameters

### ProgressParam1

If the progress window already exists: If *Param1* is the word OFF, the window is destroyed. If *Param1* is the word SHOW, the window is shown if it is currently hidden.

Otherwise, if *Param1* is a pure number, its bar's position is changed to that value. If *Param1* is blank, its bar position will be unchanged but its text will be updated to reflect any new strings provided in *SubText*, *MainText*, and *WinTitle*. In both of these modes, if the window doesn't yet exist, it will be created with the defaults for all options.

If the progress window does not exist: A new progress window is created (replacing any old one), and *Param1* is a string of zero or more options from the list below.

### SubText

The text to display below the image or bar indicator. Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`\n`). To set an existing window's text to be blank, specify `%A_Space%`. For the purpose of auto-calculating the window's height, blank lines can be reserved in a way similar to *MainText* below.

### MainText

The text to display above the image or bar indicator (its font is semi-bold). Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`\n`).

If blank or omitted, no space will be reserved in the window for *MainText*. To reserve space for single line to be added later, or to set an existing window's text to be blank, specify `%A_Space%`. To reserve extra lines beyond the first, append one or more linefeeds (`\n`).

Once the height of *MainText*'s control area has been set, it cannot be changed without recreating the window.

### WinTitle

The title of the window. If omitted and the window is being newly created, the title defaults to the name of the script (without path). If the **B**

(borderless) option has been specified, there will be no visible title bar but the window can still be referred to by this title in commands such as [WinMove](#).

### FontName

The name of the font to use for both *MainText* and *SubText*. The [font table](#) lists the fonts included with the various versions of Windows. If unspecified or if the font cannot be found, the system's default GUI font will be used.

See the options section below for how to change the size, weight, and color of the font.

## Window Size, Position, and Behavior

**A:** The window will not be always-on-top.

**B:** Borderless: The window will have no border and no title bar. To have a border but no title bar, use **B1** for a thin border or **B2** for a dialog style border.

**M:** The window will be movable by the user (except if borderless). To additionally make the window resizable (by means of dragging its borders), specify **M1**. To additionally include a system menu and a set of minimize/maximize/close buttons in the title bar, specify **M2**.

**Pn:** For Progress windows, specify for **n** the starting position of the bar (the default is 0 or the number in the allowable range that is closest to 0). To later change the position of the bar, follow this example: `Progress, 50`.

**Rx-y:** For Progress windows, specify for **x-y** the numerical range of the bar (if the **R** option is not present, the default is 0-100). For example, `R0-1000` would allow a numbers between 0 and 1000; `R-50-50` would allow numbers between -50 and 50; and `R-10--5` would allow numbers between -10 and -5.

**T:** The window will be given a button in the task bar and it will be unowned. Normally, there is no button because the window is owned by the script's main window. This setting also prevents a GUI window's `Gui +OwnDialogs` setting from taking ownership of Progress window.

**Hn:** Specify for **n** the height of the window's client area (which is the area not

including title bar and borders). If unspecified, the height will be calculated automatically based on the height of the image/bar and text in the window.

**Wn:** Specify for **n** the width of the window's client area. If unspecified, it will default to 300.

**Xn:** Specify for **n** the x-coordinate of the window's upper left corner. If unspecified, the window will be horizontally centered on the screen.

**Yn:** Specify for **n** the y-coordinate of the window's upper left corner. If unspecified, the window will be vertically centered on the screen.

**Hide:** The window will be initially hidden. Use `Progress Show` to show it later.

## Layout of Objects in the Window

**Cxy:** Centered: If this option is absent, both *SubText* and *MainText* will be centered inside the window. Specify 0 for x to make *SubText* left-justified. Specify a 1 to keep it centered. The same is true for y except that it applies to *MainText* (y can be omitted). For example: `c10`.

**ZHn:** Height of object: For Progress windows, specify for n the thickness of the progress bar (default 20). Specify -1 to make the height proportional to the width specified in ZWn (i.e. "keep aspect ratio"). If unspecified, the actual image height will be used. In either case, a value of 0 can be specified to omit the object entirely, which allows the window to be used to display only text in custom fonts, sizes, and colors.

**ZXn:** Specify for n the amount of margin space to leave on the left/right sides of the window. The default is 10.

**ZYn:** Specify for n the amount of vertical blank space to leave at the top and bottom of the window and between each control. The default is 5.

Note: To create a vertical progress bar or to have more layout flexibility, use the `Gui` command such as this example:

```
Gui, Add, Progress, Vertical vMyProgress
Gui, Show
return
; ... later...
GuiControl,, MyProgress, +10 ; Move the bar
```

upward by 10 percent. Omit the + to set an absolute position.

## Font Size and Weight

**FMn:** Specify for **n** the font size for *MainText*. Default 0, which causes 10 to be used on most systems. This default is not affected by the system's selected font size in Control Panel > Display settings.

**FSn:** Specify for **n** the font size for *SubText*. Default 0, which causes 8 to be used on most systems.

**WMn:** Specify for **n** the font weight of *MainText*. The weight should be between 1 and 1000. If unspecified, 600 (semi-bold) will be used.

**WSn:** Specify for **n** the font weight of *SubText*. The weight should be between 1 and 1000 (700 is traditionally considered to be "bold"). If unspecified, 400 (normal) will be used.

## Object Colors

A color can be one of the names from the list below or a 6-digit hexadecimal RGB value. For example, specifying `cw1A00FF` would set a window background color with red component 1A, green component 00, and blue component FF.

Add a space after each color option if there are any more options that follow it. For example: `cbRed ct900000 cwBlue`.

**CBn**: Color of progress bar: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default bar color will be used. Specify the word Default to return to the system's default progress bar color.

**CTn**: Color of text: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default text color (usually black) will be used. Specify the word Default to return to the system's default text color.

**CWn**: Color of window (background): Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's color for the face of buttons will be used (specify the word Default to later return to this color). To make the background transparent, use [WinSetTransparentColor](#).

### Color names and RGB values



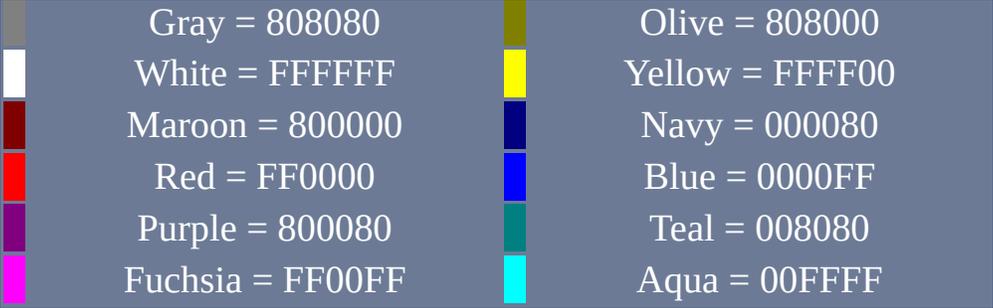
Black = 000000

Silver = C0C0C0



Green = 008000

Lime = 00FF00



|                  |                 |
|------------------|-----------------|
| Gray = 808080    | Olive = 808000  |
| White = FFFFFFFF | Yellow = FFFF00 |
| Maroon = 800000  | Navy = 000080   |
| Red = FF0000     | Blue = 0000FF   |
| Purple = 800080  | Teal = 008080   |
| Fuchsia = FF00FF | Aqua = 00FFFF   |

## Remarks

If the first parameter is the word **OFF**, the window is destroyed.

Each script can display up to 10 Progress windows. Each window has a number assigned to it at the time it is created. If unspecified, that number is 1 (the first). Otherwise, precede the first parameter with the number of the window followed by a colon. For example, the Progress command with `2:Off` would turn off the number-2 Progress window, `2:75` would set its bar to 75%, `2:` would change one or more of its text fields, and `2:B` would create a new borderless Progress window.

Upon creation, a window that would be larger than the desktop is automatically shrunk to fit.

A naked progress bar can be displayed by specifying no *SubText*, no *MainText*, and including the following options: `0 zx0 zy0`. A naked image can be displayed the same way except that only the B option is needed.

On Windows XP or later, if there is a non-Classic theme in effect, the interior of a progress bar may appear as a series of segments rather than a smooth continuous bar. To avoid this, specify a bar color explicitly such as `cbBlue`.

Use decimal (not hexadecimal) numbers for option letters that need them, except where noted.

Commands such as [WinSetAlwaysOnTop](#) and [WinMove](#) can be used to change

the attributes of an existing window without having to recreate it.

A GUI window may take ownership of a Progress window by means of `Gui +OwnDialogs`. This causes the Progress to stay always on top of its owner. In addition, the Progress is automatically destroyed when its GUI window is destroyed.

## Related

[GUI](#), [SplashImage](#), [SplashTextOn](#), [ToolTip](#)

## Examples

```
Progress, b w200, My SubText, My MainText, My
Title
Progress, 50 ; Set the position of the bar to 50%.
Sleep, 4000
Progress, Off
```

```
; Create a window just to display some 18-point
Courier text:
```

```
Progress, m2 b fs18 zh0, This is the Text.`nThis
is a 2nd line., , , Courier New
```

# SplashImage

Creates or updates a window containing an image.

```
SplashImage [ImageFile, Options, SubText, MainText,
WinTitle, FontName]
```

```
SplashImage, Off
```

```
Command Example: SplashImage "C:\My
Pictures\Company Logo.gif"
```

```
Function Example: SplashImage("C:\My
Pictures\Company Logo.gif")
```

## Parameters

### ImageFile

If this is the word OFF, the window is destroyed. If this is the word SHOW, the window is shown if it is currently hidden.

Otherwise, this is the file name of the BMP, GIF, or JPG image to display (to display other file formats such as PNG, TIF, and ICO, consider using the [Gui](#) command to create a window containing a picture control).

[AHK\_L 59+]: Any image format supported by Gui may be used with SplashImage.

*ImageFile* is assumed to be in %A\_WorkingDir% if an absolute path isn't specified. If *ImageFile* and *Options* are blank and the window already

exists, its image will be unchanged but its text will be updated to reflect any new strings provided in *SubText*, *MainText*, and *WinTitle*.

For newly created windows, if *ImageFile* is blank or there is a problem loading the image, the window will be displayed without the picture.

## Options

A string of zero or more options from the list further below.

## SubText

The text to display below the image or bar indicator. Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`\n`). To set an existing window's text to be blank, specify `%A_Space%`. For the purpose of auto-calculating the window's height, blank lines can be reserved in a way similar to *MainText* below.

## MainText

The text to display above the image or bar indicator (its font is semi-bold). Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`\n`).

If blank or omitted, no space will be reserved in the window for *MainText*. To reserve space for single line to be added later, or to set an existing window's text to be blank, specify `%A_Space%`. To reserve extra lines beyond the first, append one or more linefeeds (`\n`).

Once the height of *MainText*'s control area has been set, it cannot be

changed without recreating the window.

### WinTitle

The title of the window. If omitted and the window is being newly created, the title defaults to the name of the script (without path). If the **B** (borderless) option has been specified, there will be no visible title bar but the window can still be referred to by this title in commands such as [WinMove](#).

### FontName

The name of the font to use for both *MainText* and *SubText*. The [font table](#) lists the fonts included with the various versions of Windows. If unspecified or if the font cannot be found, the system's default GUI font will be used.

See the options section below for how to change the size, weight, and color of the font.

## Window Size, Position, and Behavior

**A:** The window will not be always-on-top.

**B:** Borderless: The window will have no border and no title bar. To have a border but no title bar, use **B1** for a thin border or **B2** for a dialog style border.

**M:** The window will be movable by the user (except if borderless). To additionally make the window resizable (by means of dragging its borders), specify **M1**. To additionally include a system menu and a set of minimize/maximize/close buttons in the title bar, specify **M2**.

**T:** The window will be given a button in the task bar and it will be unowned. Normally, there is no button because the window is owned by the script's main window. This setting also prevents a GUI window's `Gui +OwnDialogs` setting from taking ownership of a Splash window.

**Hn:** Specify for **n** the height of the window's client area (which is the area not including title bar and borders). If unspecified, the height will be calculated automatically based on the height of the image/bar and text in the window.

**Wn:** Specify for **n** the width of the window's client area. If unspecified, the width will be calculated automatically for `SplashImage` if there is an image. Otherwise, it will default to 300.

**Xn:** Specify for **n** the x-coordinate of the window's upper left corner. If unspecified, the window will be horizontally centered on the screen.

**Yn:** Specify for **n** the y-coordinate of the window's upper left corner. If unspecified, the window will be vertically centered on the screen.

**Hide:** The window will be initially hidden. Use `SplashImage Show` to show it later.

## Layout of Objects in the Window

**Cxy:** Centered: If this option is absent, both *SubText* and *MainText* will be centered inside the window. Specify 0 for x to make *SubText* left-justified. Specify a 1 to keep it centered. The same is true for y except that it applies to *MainText* (y can be omitted). For example: `c10`.

**ZHn:** Height of object: Specify for n the height to which to scale image. Specify -1 to make the height proportional to the width specified in ZWn (i.e. "keep aspect ratio"). If unspecified, the actual image height will be used. In either case, a value of 0 can be specified to omit the object entirely, which allows the window to be used to display only text in custom fonts, sizes, and colors.

**ZWn:** Width of object: Specify for n the width to which to scale the image. Specify -1 to make the width proportional to the height specified in ZHn (i.e. "keep aspect ratio"). If unspecified, the actual image width will be used.

**ZXn:** Specify for n the amount of margin space to leave on the left/right sides of the window. The default is 0.

**ZYn:** Specify for n the amount of vertical blank space to leave at the top and bottom of the window and between each control. The default is 0.

## Font Size and Weight

**FMn:** Specify for **n** the font size for *MainText*. Default 0, which causes 10 to be used on most systems. This default is not affected by the system's selected font size in Control Panel > Display settings.

**FSn:** Specify for **n** the font size for *SubText*. Default 0, which causes 8 to be used on most systems.

**WMn:** Specify for **n** the font weight of *MainText*. The weight should be between 1 and 1000. If unspecified, 600 (semi-bold) will be used.

**WSn:** Specify for **n** the font weight of *SubText*. The weight should be between 1 and 1000 (700 is traditionally considered to be "bold"). If unspecified, 400 (normal) will be used.

## Object Colors

A color can be one of the names from the list below or a 6-digit hexadecimal RGB value. For example, specifying `cw1A00FF` would set a window background color with red component 1A, green component 00, and blue component FF.

Add a space after each color option if there are any more options that follow it. For example: `cbRed ct900000 cwBlue`.

**CTn:** Color of text: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default text color (usually black) will be used. Specify the word Default to return to the system's default text color.

**CWn:** Color of window (background): Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's color for the face of buttons will be used (specify the word Default to later return to this color). To make the background transparent, use [WinSetTransparentColor](#).

### Color names and RGB values

|                                                                                     |                  |                                                                                     |                 |
|-------------------------------------------------------------------------------------|------------------|-------------------------------------------------------------------------------------|-----------------|
|  | Black = 000000   |  | Green = 008000  |
|  | Silver = C0C0C0  |  | Lime = 00FF00   |
|  | Gray = 808080    |  | Olive = 808000  |
|  | White = FFFFFFFF |  | Yellow = FFFF00 |
|  | Maroon = 800000  |  | Navy = 000080   |
|  | Red = FF0000     |  | Blue = 0000FF   |
|  | Purple = 800080  |  | Teal = 008080   |
|  | Fuchsia = FF00FF |  | Aqua = 00FFFF   |

## Remarks

If the first parameter is the word **OFF**, the window is destroyed.

Each script can display up to 10 SplashImage windows. Each window has a number assigned to it at the time it is created. If unspecified, that number is 1 (the first). Otherwise, precede the first parameter with the number of the window followed by a colon. For example, the SplashImage command with `2:Off` would turn off the number-2 SplashImage window, `2:` would change one or more of its text fields, and `2:C:\My Images\Picture1.jpg` would create a new number-2 SplashImage window.

Upon creation, a window that would be larger than the desktop is automatically shrunk to fit.

A naked image can be displayed the same way except that only the B option is needed.

Use decimal (not hexadecimal) numbers for option letters that need them, except where noted.

Commands such as [WinSetAlwaysOnTop](#) and [WinMove](#) can be used to change the attributes of an existing window without having to recreate it.

A GUI window may take ownership of a Splash window by means of [Gui +OwnDialogs](#). This causes the SplashImage to stay always on top of its owner. In addition, the SplashImage is automatically destroyed when its GUI window is

destroyed.

## Related

[GUI](#), [Progress](#), [SplashTextOn](#), [ToolTip](#)

## Examples

**; Create a simple SplashImage window:**

```
SplashImage, C:\My Pictures\Company Logo.gif
```

**; Create a borderless SplashImage window with some large text beneath the image:**

```
SplashImage, C:\My Pictures\Company Logo.gif, b
fs18, This is our company logo.
Sleep, 4000
SplashImage, Off
```

**; Here is a working example that demonstrates how a Progress window can be**

**; overlaid on a SplashImage to make a professional looking Installer screen:**

```
IfExist, C:\WINDOWS\system32\ntimage.gif,
SplashImage, %A_WinDir%\system32\ntimage.gif, A,,,
Installation
Loop, %A_WinDir%\system32*.*
{
 Progress, %a_index%, %a_loopfilename%,
Installing..., Draft Installation
 Sleep, 50
 if a_index = 100
 break
}
```

**; There is similar example at the bottom of the GUI page. Its advantage is that it uses only a single**

**; window and it gives you more control over window layout.**

# SplashTextOn / SplashTextOff

Creates a customizable text popup window.

## SplashTextOff

```
SplashTextOn [, Width, Height, Title, Text]
```

**Command Example:** `SplashTextOn 200, 150, MyTitle, MyText`

**Function Example:** `SplashTextOn(200, 150, "MyTitle", "MyText")`

## Parameters

### Width

The width in pixels of the Window. Default 200. This parameter can be an [expression](#).

### Height

The height in pixels of the window (not including its title bar). Default 0 (i.e. just the title bar will be shown). This parameter can be an [expression](#).

### Title

The title of the window. Default empty (blank).

### Text

The text of the window. Default empty (blank). If *Text* is long, it can be

broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

## Remarks

To have more control over layout and font name/color/size, use the [Progress](#) command with the `zh0` option, which omits the bar and displays only text. For example: `Progress, zh0 fs18, Some 18-point text to display.`

Use the `SplashTextOff` command to remove an existing splash window.

The splash window is "always on top", meaning that it stays above all other normal windows. To change this, use `WinSetAlwaysOnTop, Off, <insert title of splash window>`. [WinSetTransparent](#) can also make the splash window transparent.

[WinMove](#) can be used to reposition and resize the `SplashText` window after it has been displayed with this command.

Unlike [Progress](#), [SplashImage](#), [MsgBox](#), [InputBox](#), [FileSelectFile](#), and [FileSelectFolder](#), only one `SplashText` window per script is possible.

If `SplashTextOn` is used while the splash window is already displayed, the window will be recreated with the new parameter values. However, rather than recreating the splash window every time you wish to change its title or text, better performance can be achieved by using the following, especially if the window needs to be changed frequently:

```
WinSetTitle, <insert title of splash window>, ,
NewTitle
ControlSetText, Static1, NewText, <insert title
of splash window>
```

## Related

[Progress](#), [SplashImage](#), [ToolTip](#), [MsgBox](#), [InputBox](#), [FileSelectFile](#),  
[FileSelectFolder](#), [WinMove](#)

## Example

```
SplashTextOn, , , Displays only a title bar.
Sleep, 2000
SplashTextOn, 400, 300, Clipboard, The clipboard
contains:`n%clipboard%
WinMove, Clipboard, , 0, 0 ; Move the splash
window to the top left corner.
Msgbox, Press OK to dismiss the SplashText
SplashTextOff
```

# ToolTip

Creates an always-on-top window anywhere on the screen.

```
ToolTip [Text, X, Y, WhichToolTip]
```

```
Command Example: ToolTip "Text"
Function Example: ToolTip("Text")
```

## Parameters

### Text

If blank or omitted, the existing tooltip (if any) will be hidden. Otherwise, this parameter is the text to display in the tooltip. To create a multi-line tooltip, use the linefeed character (`\n`) in between each line, e.g.

Line1\nLine2.

If *Text* is long, it can be broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

### X, Y

The X and Y position of the tooltip relative to the active window (use [CoordMode](#), `ToolTip` to change to screen coordinates). If the coordinates are omitted, the tooltip will be shown near the mouse cursor.

### WhichToolTip

Omit this parameter if you don't need multiple tooltips to appear simultaneously. Otherwise, this is a number between 1 and 20 to indicate which tooltip window to operate upon. If unspecified, that number is 1 (the first).

## Remarks

If the X & Y coordinates would cause the tooltip to run off screen, it is repositioned to be entirely visible.

The tooltip is displayed until one of the following occurs:

- The script terminates.
- The ToolTip command is executed again with a blank *Text* parameter.
- The user clicks on the tooltip (this behavior may vary depending on operating system version).

A GUI window may be made the owner of a tooltip by means of `Gui +OwnDialogs`. Such a tooltip is automatically destroyed when its owner is destroyed.

## Related

[CoordMode](#), [TrayTip](#), [GUI](#), [MsgBox](#), [InputBox](#), [FileSelect](#), [DirSelect](#)

## Example

```
ToolTip, Multiline`nTooltip, 100, 150
```

```
; To have a ToolTip disappear after a certain
amount of time
; without having to use Sleep (which stops the
current thread):
```

```
ToolTip, Timed ToolTip`nThis will be displayed for
5 seconds.
```

```
SetTimer, RemoveToolTip, -5000
return
```

```
RemoveToolTip:
ToolTip
return
```

# TrayTip

Creates a balloon message window near the tray icon. On Windows 10, a toast notification may be shown instead.

```
TrayTip [Text, Title, Options]
```

```
Command Example: TrayTip "Message from
AutoHotkey", "AutoHotkey"
```

```
Function Example: TrayTip("Message from
AutoHotkey", "AutoHotkey")
```

## Parameters

### Text

The message to display. Only the first 265 characters will be displayed.

Carriage return (`\r`) or linefeed (`\n`) may be used to create multiple lines of text. For example: `Line1\nLine2`.

If *Text* is long, it can be broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

It is possible to show a window with only a title by leaving *Text* blank.

### Title

The title of the window. Only the first 73 characters will be displayed.

If *Title* is blank, the title line will be entirely omitted from the window, making it vertically shorter.

## Options

Either an integer value (a combination by addition or bitwise-OR) or a string of zero or more case-insensitive options separated by at least one space or tab. One or more numeric options may also be included in the string.

| Function                                                    | String Value | Decimal Value | Hex Value |
|-------------------------------------------------------------|--------------|---------------|-----------|
| Info icon                                                   | Iconi        | 1             | 0x1       |
| Warning icon                                                | Icon!        | 2             | 0x2       |
| Error icon                                                  | Iconx        | 3             | 0x3       |
| Windows XP and later: Do not play the notification sound.   | Mute         | 16            | 0x10      |
| Windows Vista and later: Use the large version of the icon. |              | 32            | 0x20      |

If omitted, it defaults to 0, which is no icon. The icon is also not shown by the balloon window if it lacks a *Title* (this does not apply to Windows 10 toast notifications).

## To Hide the Window

To hide a TrayTip balloon window, omit both *Text* and *Title*. For example:

```
TrayTip
```

To hide a Windows 10 toast notification, temporarily remove the tray icon. For example:

```
TrayTip #1, This is TrayTip #1
Sleep 3000 ; Let it display for 3 seconds.
HideTrayTip()
TrayTip #2, This is the second notification.
Sleep 3000

; Copy this function into your script to use
it.
HideTrayTip() {
 TrayTip ; Attempt to hide it the normal
way.
 if SubStr(A_OSVersion,1,3) = "10." {
 Menu Tray, NoIcon
 Sleep 200 ; It may be necessary to
adjust this sleep.
 Menu Tray, Icon
 }
}
```

This may not always work, according to at least one report.

## Remarks

**Windows 10** replaces all balloon windows with toast notifications by default (this can be overridden via group policy). Calling TrayTip multiple times will usually cause multiple notifications to be placed in a "queue" instead of each notification replacing the last.

TrayTip has no effect if the script lacks a tray icon (via `#NoTrayIcon` or `Menu` 

Tray, NoIcon). TrayTip also has no effect if the following REG\_DWORD value exists and has been set to 0:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\
Explorer\Advanced >> EnableBalloonTips
```

On a related note, there is a tooltip displayed whenever the user hovers the mouse over the script's tray icon. The contents of this tooltip can be changed via: `Menu, Tray, Tip, My New Text`.

## Related

[ToolTip](#), [SetTimer](#), [Menu](#), [MsgBox](#), [InputBox](#), [FileSelect](#), [DirSelect](#)

## Examples

```
TrayTip, Multiline`nText, My Title, Icon1 Mute
```

```
; To have more precise control over the display
time without
; having to use Sleep (which stops the current
thread):
```

```
#Persistent
TrayTip, This will be displayed for 5 seconds.,
Timed TrayTip
SetTimer, HideTrayTip, -5000
```

```
HideTrayTip() { ; NOTE: For Windows 10, replace
this function with the one defined above.
```

```
 TrayTip
}
```

---

```
; To have a TrayTip permanently displayed, use a
timer to refresh it periodically.
```

```
; NOTE: This probably won't work well on Windows
10 for reasons described above.
```

```
#Persistent
```

```
SetTimer, RefreshTrayTip, 1000
```

```
Gosub, RefreshTrayTip ; Call it once to get it
started right away.
```

```
return
```

```
RefreshTrayTip:
```

```
TrayTip, This is a more permanent TrayTip.,
```

```
Refreshed TrayTip, Mute
```

```
return
```

# Math Functions

**Note:** Math functions generally return a blank value (empty string) if any of the incoming parameters are non-numeric, or the operation is invalid (such as divide by zero).

## Quick Reference:

- **General Math:**
  - Abs - Absolute value
  - Ceil - Rounding up
  - Exp - Exponential
  - Floor - Rounding down
  - Log - Decimal logarithm
  - Ln - Natural logarithm
  - Mod - Remainder after division (Modulo)
  - Round - Rounding
  - Sqrt - Square root
- **Trigonometry:**
  - Sin - Sine
  - Cos - Cosine
  - Tan - Tangent
  - ASin - Arcsine
  - ACos - Arccosine
  - ATan - Arctangent
- **Error-handling**

# General Math

## **Abs** Absolute value

```
Value := Abs(Number)
```

Returns the absolute value of *Number*.

The return value is the same type as *Number* (integer or floating point).

```
MsgBox, % Abs(-1.2) ; Returns 1.2
```

## Ceil Rounding up

```
Value := Ceil(Number)
```

Returns *Number* rounded up to the nearest integer (without any .00 suffix).

```
MsgBox, % Ceil(1.2) ; Returns 2
MsgBox, % Ceil(-1.2) ; Returns -1
```

## **Exp** Exponential

```
Value := Exp(N)
```

Returns  $e$  (which is approximately 2.71828182845905) raised to the  $N$ th power.

$N$  may be negative and may contain a decimal point. To raise numbers other than  $e$  to a power, use the `**` operator.

```
MsgBox, % Exp(1.2) ; Returns 3.320117
```

## **Floor** Rounding down

```
Value := Floor(Number)
```

Returns *Number* rounded down to the nearest integer (without any .00 suffix).

```
MsgBox, % Floor(1.2) ; Returns 1
MsgBox, % Floor(-1.2) ; Returns -2
```

## Log Decimal logarithm

```
Value := Log(Number)
```

Returns the logarithm (base 10) of *Number*.

The result is a floating-point number. If *Number* is negative, an empty string is returned.

```
MsgBox, % Log(1.2) ; Returns 0.079181
```

## **Ln** Natural logarithm

```
Value := Ln(Number)
```

Returns the natural logarithm (base e) of *Number*.

The result is a floating-point number. If *Number* is negative, an empty string is returned.

```
MsgBox, % Ln(1.2) ; Returns 0.182322
```

## **Mod**    Remainder after division (Modulo)

```
Value := Mod(Dividend, Divisor)
```

Returns the remainder when *Dividend* is divided by *Divisor*.

The sign of the result is always the same as the sign of the first parameter. If either input is a floating point number, the result is also a floating point number. If the second parameter is zero, the function yields a blank result (empty string).

```
MsgBox, % Mod(7.5, 2) ; Returns 1.5 (2 x 3 +
1.5)
```

## Round Rounding

```
Value := Round(Number [, N])
```

Returns *Number* rounded to *N* decimal places.

If *N* is omitted or 0, *Number* is rounded to the nearest integer:

```
MsgBox, % Round(3.14) ; Returns 3
```

If *N* is positive number, *Number* is rounded to *N* decimal places:

```
MsgBox, % Round(3.14, 1) ; Returns 3.1
```

If *N* is negative, *Number* is rounded by *N* digits to the left of the decimal point:

```
MsgBox, % Round(345, -1) ; Returns 350
MsgBox, % Round(345, -2) ; Returns 300
```

The result is an integer if *N* is omitted or less than 1. Otherwise, the result is a numeric string with exactly *N* decimal places. If a pure number is needed, simply perform another math operation on Round()'s return value; for example: `Round(3.333, 1)+0`.

## Sqrt Square root

```
Value := Sqrt(Number)
```

Returns the square root of *Number*.

The result is a floating-point number. If *Number* is negative, the function yields a blank result (empty string).

```
MsgBox, % Sqrt(16) ; Returns 4
```

## Trigonometry

**Note:** To convert a radians value to degrees, multiply it by  $180/\pi$  (approximately 57.29578). To convert a degrees value to radians, multiply it by  $\pi/180$  (approximately 0.01745329252). The value of  $\pi$  (approximately 3.141592653589793) is 4 times the arctangent of 1.

## **Sin** Sine

```
Value := Sin(Number)
```

Returns the trigonometric sine of *Number*.

*Number* must be expressed in radians.

```
MsgBox, % Sin(1.2) ; Returns 0.932039
```

## Cos Cosine

```
Value := Cos(Number)
```

Returns the trigonometric cosine of *Number*.

*Number* must be expressed in radians.

```
MsgBox, % Cos(1.2) ; Returns 0.362358
```

## **Tan** Tangent

```
Value := Tan(Number)
```

Returns the trigonometric tangent of *Number*.

*Number* must be expressed in radians.

```
MsgBox, % Tan(1.2) ; Returns 2.572152
```

## **ASin**    Arcsine

```
Value := ASin(Number)
```

Returns the arcsine (the number whose sine is *Number*) in radians.

If *Number* is less than -1 or greater than 1, the function yields a blank result (empty string).

```
MsgBox, % ASin(0.2) ; Returns 0.201358
```

## **ACos** Arccosine

```
Value := ACos(Number)
```

Returns the arccosine (the number whose cosine is *Number*) in radians.

If *Number* is less than -1 or greater than 1, the function yields a blank result (empty string).

```
MsgBox, % ACos(0.2) ; Returns 1.369438
```

## **ATan** Arctangent

```
Value := ATan(Number)
```

Returns the arctangent (the number whose tangent is *Number*) in radians.

```
MsgBox, % ATan(1.2) ; Returns 0.876058
```

## Error-Handling

These functions return a blank result (empty string) if any incoming parameters are non-numeric or an invalid operation (such as divide by zero) is attempted.

# DateAdd

Adds or subtracts time from a [date-time](#) value.

```
OutputVar := DateAdd(DateTime, Time, TimeUnits)
```

```
Function Example: Yesterday :=
DateAdd(A_Now, -1, "days")
```

## Parameters

### DateTime

A date-time stamp in the [YYYYMMDDHH24MISS](#) format.

### Time

The amount of time to add, as an integer or floating-point number. Specify a negative number to perform subtraction.

### TimeUnits

The meaning of the *Time* parameter. *TimeUnits* may be one of the following strings (or just the first letter):  
Seconds, Minutes, Hours or Days.

## Return Value

If *DateTime* contains an invalid timestamp or a year prior to 1601, or if *Time* is non-numeric, an empty string is returned to indicate the error.

Otherwise, a timestamp in [YYYYMMDDHH24MISS](#) format is returned.

## Remarks

The built-in variable **A\_Now** contains the current local time in [YYYYMMDDHH24MISS](#) format.

To calculate the amount of time between two timestamps, use [DateDiff](#).

## Related

[DateDiff](#), [FileGetTime](#), [FormatTime](#)

## Example

```
; Calculate the date 31 days from now.
later := DateAdd(A_Now, 31, "days")
MsgBox % FormatTime(later)
```

# DateDiff

Compares two [date-time](#) values and returns the difference.

```
OutputVar := DateDiff(DateTime1, DateTime2,
TimeUnits)
```

```
Function Example: Days := DateDiff("20100101",
A_Now, "days")
```

## Parameters

**DateTime1,**

**DateTime2**

Date-time stamps in the [YYYYMMDDHH24MISS](#) format.

If *DateTime1* is earlier than *DateTime2*, the result is a negative number.

**TimeUnits**

Units to measure the difference in. *TimeUnits* may be one of the following strings (or just the first letter):

Seconds, Minutes, Hours or Days.

## Return Value

If *DateTime* contains an invalid timestamp or a year prior to 1601, an empty string is returned to indicate the error.

Otherwise, the return value is the difference between the two timestamps, in the units specified by *TimeUnits*. The result is always rounded *down* to the nearest integer. For example, if the actual difference between two timestamps is 1.999 days, it will be reported as 1 day. If higher precision is needed, specify *Seconds* for *TimeUnits* and divide the result by 60.0, 3600.0, or 86400.0.

## Remarks

The built-in variable **A\_Now** contains the current local time in `YYYYMMDDHH24MISS` format.

To precisely determine the elapsed time between two events, use the `A_TickCount` method because it provides millisecond precision.

To add or subtract a certain number of seconds, minutes, hours, or days from a timestamp, use `DateAdd` (subtraction is achieved by adding a negative number).

## Related

`DateAdd`, `FileGetTime`, `FormatTime`

## Example

```
var1 := "20050126"
var2 := "20040126"
diff := DateDiff(var1, var2, "days")
MsgBox %diff% ; The answer will be 366 since 2004
is a leap year.
```

# Random

Generates a pseudo-random number.

```
OutputVar := Random(Min, Max)
RandomSeed()
```

```
Function Example: rand := Random(10, 100)
```

## Parameters

### Min

The smallest number that can be generated, which can be negative or floating point. If omitted, the smallest number will be 0. The lowest allowed value is -2147483648 for integers, but floating point numbers have no restrictions.

### Max

The largest number that can be generated, which can be negative or floating point. If omitted, the largest number will be 2147483647 (which is also the largest allowed integer value -- but floating point numbers have no restrictions).

## Remarks

This function returns a pseudo-randomly generated number, which is a number that simulates a true random number but is really a number based on a complicated formula to make determination/guessing of the next number extremely difficult.

All numbers within the specified range have approximately the same probability of being generated (however, see "known limitations" below).

If either *Min* or *Max* contains a decimal point, the end result will be a floating point number. Otherwise, the result will be an integer.

Known limitations for floating point: 1) only about 4,294,967,296 distinct numbers can be generated for any particular range, so all other numbers in the range will never be generated; 2) occasionally a result can be slightly greater than the specified *Max* (this is caused in part by the imprecision inherent in floating point numbers).

## RandomSeed

Reseeds the random number generator with *NewSeed*.

**RandomSeed** *NewSeed*

This affects all subsequently generated random numbers. *NewSeed* should be an integer between 0 and 4294967295 (0xFFFFFFFF). Reseeding can improve the quality/security of generated random numbers, especially when *NewSeed* is a genuine random number rather than one of lesser quality such as a pseudo-

random number. Generally, reseeding does not need to be done more than once.

If reseeding is never done by the script, the seed starts off as the low-order 32-bits of the 64-bit value that is the number of 100-nanosecond intervals since January 1, 1601. This value travels from 0 to 4294967295 every ~7.2 minutes.

## Examples

```
number := Random(1, 10)
fraction := Random(0.0, 1.0)
```

## Comments based on the original source

This function uses the Mersenne Twister random number generator, MT19937, written by Takuji Nishimura and Makoto Matsumoto, Shawn Cokus, Matthe Bellew and Isaku Wada.

The Mersenne Twister is an algorithm for generating random numbers. It was designed with consideration of the flaws in various other generators. The period,  $2^{19937}-1$ , and the order of equidistribution, 623 dimensions, are far greater. The generator is also fast; it avoids multiplication and division, and it benefits from caches and pipelines. For more information see the inventors' web page at [www.math.keio.ac.jp/~matumoto/emt.html](http://www.math.keio.ac.jp/~matumoto/emt.html)

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, All rights reserved.

Redistribution and use in source and binary forms, with or without modification,

are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Do NOT use for CRYPTOGRAPHY without securely hashing several returned values together, otherwise the generator state can be learned after reading 624 consecutive values.

When you use this, send an email to: [matumoto@math.keio.ac.jp](mailto:matumoto@math.keio.ac.jp) with an appropriate reference to your work. It would be nice to CC: [rjwagner@writeme.com](mailto:rjwagner@writeme.com) and [Cokus@math.washington.edu](mailto:Cokus@math.washington.edu) when you write.

*This above has been already been done for AutoHotkey, but if you use the Random command in a publicly distributed application, consider sending an e-mail to the above people to thank them.*

# MemoryModule

## Table of Contents

- [About Memory Module](#)
- [Load a dll multiple times](#)
- [COM Support](#)
- [Built-in functions.](#)

## About Memory Module

[MemoryModule](#) was developed by Joachim Bauch and is released under MPL 2.0.

The default windows API functions to load external libraries into a program ([LoadLibrary](#), [LoadLibraryEx](#)) only work with files on the filesystem.

[MemoryModule](#) is a library that can be used to load a DLL completely from memory - without storing on the disk first.

**Load a [dll](#) multiple times:** [AutoHotkey](#) is not designed for multi-threading naturally and most likely it will be never implemented. Using [AutoHotkey.dll](#) we can still run multiple scripts in one process, on a multi-core system we can even run multiple scripts at the same time. To do so, [AutoHotkey](#) module also needs to be loaded multiple times. Instead of using multiple dlls by copying and renaming them we can use [Memory Module](#).

**COM support:** [AutoHotkey\\_H](#) additionally supports loading unregistered dlls using [ComObjDll](#).

Internally [AutoHotkey.dll](#) COM Interface always uses [MemoryModule](#) to create a new thread, this allows loading the same module multiple times.

[AutoHotkey.dll](#) automatically frees the module when COM object is released.

**Functions:** Following functions are available

- [MemoryLoadLibrary](#) - Load a [dll](#) from Memory, similar to [LoadLibrary](#).
- [MemoryGetProcAddress](#) - Get address of a function in [dll](#), similar to

## GetProcAddress

- [MemoryFreeLibrary](#) - Free dll, similar to [FreeLibrary](#)
- [MemoryFindResource](#) - Find Resource in a dll, similar to [FindResource](#) and [FindResourceEx](#)
- [MemorySizeofResource](#) - Retrieves the size in bytes, of the specified resource in a dll, similar to [SizeOfResource](#)
- [MemoryLoadResource](#) - Load the specified resource in a dll, similar to [LoadResource](#)
- [MemoryLoadString](#) - Load the specified String resource in a dll, similar to [LoadString](#)

# MemoryLoadLibrary

Loads the specified [dll](#) into the process. Similar to [LoadLibrary](#) but loads the module from memory rather than disk and allows loading a module multiple times.

```
OutputVar := MemoryLoadLibrary(PathToDll)
```

```
Function Example: ahkdllModule :=
MemoryLoadLibrary(A_AhkDir "\AutoHotkey.dll")
```

## Parameters

### OutputVar

The name of the variable in which to store the handle of loaded module.

### PathToDll

Path to a [dll](#) file saved on disk or network.

## Related

[MemoryModule](#), [MemoryGetProcAddress](#), [MemoryFreeLibrary](#),  
[MemoryFindResource](#), [MemorySizeofResource](#), [MemoryLoadResource](#),  
[MemoryLoadString](#), [DllCall](#), [DynaCall](#)

## Examples

```
MemoryLoadLibrary, ahkdllModule,
%A_ScriptDir%\AutoHotkey.dll ; Load the AutoHotkey
module.
MemoryGetProcAddress, ahkdll, %ahkdllModule%,
ahkdll ; get address of ahkdll function.
MemoryGetProcAddress, ahkReady, %ahkdllModule%,
ahkReady ; get address of ahkReady function.
DllCall(ahkdll, "Str", "NewScript.ahk", "Str", "") ;
start new thread
While DllCall(ahkReady)
 Sleep 100
MemoryFreeLibrary(ahkdllModule)
```

# MemoryGetProcAddress

Find the function pointer in the specified [dll](#) previously loaded with [MemoryLoadLibrary](#). Similar to [GetProcAddress](#).

```
OutputVar := MemoryGetProcAddress(Handle, FuncName)
```

```
Function Example: ahkdll :=
MemoryGetProcAddress(ahkdllModule, "ahkdll")
```

## Parameters

### OutputVar

The name of the variable in which to store the function pointer.

### Handle

MemoryModule handle previously returned by [MemoryLoadLibrary](#), or [ResourceLoadLibrary](#).

### FuncName

Name of the function.

## Related

[MemoryModule](#), [MemoryLoadLibrary](#), [ResourceLoadLibrary](#),  
[MemoryFreeLibrary](#), [MemoryFindResource](#), [MemorySizeofResource](#),

[MemoryLoadResource](#), [MemoryLoadString](#), [DllCall](#), [DynaCall](#)

## Examples

See [MemoryLoadLibrary](#)

# MemoryFreeLibrary

Free the specified [dll](#) previously loaded with [MemoryLoadLibrary](#). Similar to [FreeLibrary](#).

**MemoryFreeLibrary** Handle

```
Command Example: MemoryFreeLibrary
ahkdllModule
Function Example:
MemoryFreeLibrary(ahkdllModule)
```

## Parameters

### Handle

MemoryModule handle previously returned by [MemoryLoadLibrary](#).

## Related

[MemoryModule](#), [MemoryGetProcAddress](#), [MemoryLoadLibrary](#),  
[ResourceLoadLibrary](#), [MemoryFindResource](#), [MemorySizeofResource](#),  
[MemoryLoadResource](#), [MemoryLoadString](#), [DllCall](#), [DynaCall](#)

## Examples

See [MemoryLoadLibrary](#)

# MemoryFindResource

Find the resource in the specified [dll](#) previously loaded with [MemoryLoadLibrary](#). Similar to [FindResource](#) and [FindResourceEx](#).

```
OutputVar := MemoryFindResource(Handle, Name, Type [, Language])
```

```
Function Example: ahkdllTypeLib :=
MemoryFindResource(ahkdllModule, 1, "TYPELIB")
```

## Parameters

### OutputVar

The name of the variable in which to store the resource pointer.

### Handle

MemoryModule handle previously loaded with [MemoryLoadLibrary](#).

### Name

Name of the resource, can be a string or digit.

### Type

Type of the resource, can be a string or digit. See also [Resource Types](#).

### Language (optional)

The language of the resource.

## Related

[MemoryModule](#), [MemoryLoadLibrary](#), [ResourceLoadLibrary](#),  
[MemoryFreeLibrary](#), [MemoryGetProcAddress](#), [MemorySizeofResource](#),  
[MemoryLoadResource](#), [MemoryLoadString](#), [MemoryLoadStringEx](#), [DllCall](#),  
[DynaCall](#)

# MemorySizeOfResource

Find out the size of resource in the specified [dll](#) previously loaded with [MemoryLoadLibrary](#). Similar to [SizeOfResource](#).

```
OutputVar := MemorySizeOfResource(Handle, hResource)
```

```
Function Example: sz :=
MemorySizeOfResource(ahkdllModule, hResource)
```

## Parameters

### OutputVar

The name of the variable in which to store the function pointer.

### Handle

MemoryModule handle previously returned by [MemoryLoadLibrary](#).

### hResource

Resource handle previously returned by [MemoryFindResource](#)

## Related

[MemoryModule](#), [ResourceLoadLibrary](#), [MemoryLoadLibrary](#),  
[MemoryFreeLibrary](#), [MemoryFindResource](#), [MemorySizeOfResource](#),  
[MemoryLoadResource](#), [MemoryLoadString](#), [DllCall](#), [DynaCall](#)

## Examples

```
ahk:=MemoryLoadLibrary(A_AhkDir
"\AutoHotkey.dll")
hRes:=MemoryFindResource(ahk,"TYPELIB",1)
MsgBox % MemorySizeOfResource(ahk,hRes)
```

# MemoryLoadResource

Load a resource in the specified `dll` previously loaded with `MemoryLoadLibrary`. Similar to `LoadResource`.

```
OutputVar := MemoryLoadResource(Handle, hResource)
```

```
Function Example: DataPTR :=
MemoryLoadResource(ahkdllModule, hResource)
```

## Parameters

### OutputVar

The name of the variable in which to store the resource pointer. Note, we do not need to call `LockResource` function, `MemoryLoadResource` returns already the pointer to resource.

### Handle

MemoryModule handle previously returned by `MemoryLoadLibrary`.

### hResource

Resource handle previously returned by `MemoryFindResource`

## Related

`MemoryModule`, `MemoryLoadLibrary`, `ResourceLoadLibrary`,

MemoryFreeLibrary, MemoryFindResource, MemorySizeofResource,  
MemoryLoadResource, MemoryLoadString, DllCall, DynaCall

## Examples

```
ahk:=MemoryLoadLibrary(A_AhkDir
"\AutoHotkey.dll")
hRes:=MemoryFindResource(ahk,"TYPELIB",1)
DataPTR := MemoryLoadResource(ahk,hRes)
```

# MemoryLoadString

Loads a string resource in the specified `dll` previously loaded with `MemoryLoadLibrary`. Similar to `LoadString`.

```
OutputVar := MemoryLoadString(Handle, Id [,
BufferPointer, Length])
```

```
Function Example: StringLength :=
MemoryLoadString(ahkdllModule, 2, &var, 10)
```

## Parameters

### OutputVar

The name of the variable in which to store the string pointer if BufferPointer is omitted or 0 or size of the string if BufferPointer parameter is used.

### Handle

MemoryModule handle previously returned by `MemoryLoadLibrary`.

### Id

Id of string resource, must be a digit.

### BufferPointer (optional)

Pointer to the buffer receiving the string.

### Length (optional)

The length of string in characters.

### Language (optional)

The language of string, uses DEFAULT\_LANGUAGE by default.

## Remarks

If the BufferPointer is 0 or omitted the string from resource is returned, otherwise the string is copied to the buffer.

## Related

[MemoryModule](#), [ResourceLoadLibrary](#), [MemoryLoadLibrary](#),  
[MemoryFreeLibrary](#), [MemoryGetProcAddress](#), [MemoryFindResource](#),  
[MemorySizeofResource](#), [MemoryLoadResource](#), [DllCall](#), [DynaCall](#)

## Examples

```
lib:=MemoryLoadLibrary(A_ScriptDir
"\LiteZip.dll")
MsgBox % StrGet(MemoryLoadString(lib,1))
```

# ResourceLoadLibrary

Loads the specified [dll](#) from resources into the process. Similar to [MemoryLoadLibrary](#).

```
OutputVar := ResourceLoadLibrary(ResName)
```

```
Function Example: ahkdllModule :=
ResourceLoadLibrary("AutoHotkey.dll")
```

## Parameters

### OutputVar

The name of the variable in which to store the handle of loaded module.

### ResName

Name of RCDATA resource, created by [FileInstall](#).

## Related

[MemoryModule](#), [MemoryGetProcAddress](#), [MemoryLoadLibrary](#),  
[MemoryFreeLibrary](#), [MemoryFindResource](#), [MemorySizeofResource](#),  
[MemoryLoadResource](#), [MemoryLoadString](#), [DllCall](#), [DynaCall](#)

## Examples

```
MemoryLoadLibrary, ahkdllModule,
```

```
%A_ScriptDir%\AutoHotkey.dll ; Load the AutoHotkey
module.
MemoryGetProcAddress, ahkdll, %ahkdllModule%,
ahkdll ; get address of ahkdll function.
MemoryGetProcAddress, ahkReady, %ahkdllModule%,
ahkReady ; get address of ahkReady function.
DllCall(ahkdll,"Str","NewScript.ahk","Str","") ;
start new thread
While DllCall(ahkReady)
 Sleep 100
MemoryFreeLibrary(ahkdllModule)
```

# #HotkeyInterval

Along with [#MaxHotkeysPerInterval](#), specifies the rate of [hotkey](#) activations beyond which a warning dialog will be displayed.

**#HotkeyInterval** Milliseconds

## Parameters

### Milliseconds

The length of the interval in milliseconds.

## Remarks

If this directive is unspecified in the script, it will behave as though set to 2000.

For details and remarks, see [#MaxHotkeysPerInterval](#).

## Related

[#MaxHotkeysPerInterval](#)

## Example

```
#HotkeyInterval 2000 ; This is the default value
(milliseconds).
#MaxHotkeysPerInterval 200
```

# #HotkeyModifierTimeout

Affects the behavior of [hotkey](#) modifiers: CTRL, ALT, WIN, and SHIFT.

**#HotkeyModifierTimeout** Milliseconds

## Parameters

### Milliseconds

The length of the interval in milliseconds. The value can be -1 so that it never times out (modifier keys are always pushed back down after the Send), or 0 so that it always times out (modifier keys are never pushed back down).

## Remarks

This directive **need not** be used when:

- Hotkeys send their keystrokes via the [SendInput](#) or [SendPlay](#) methods. This is because those methods postpone the user's physical pressing and releasing of keys until after the Send completes.
- The script has the keyboard hook installed (you can see if your script uses the hook via the "View->Key history" menu item in the main window, or via the [KeyHistory](#) command). This is because the hook can keep track of which modifier keys (ALT/CTRL/WIN/SHIFT) the user is physically holding down and doesn't need to use the timeout.

To illustrate the effect of this directive, consider this example: `!a : Send, abc.`

When the `Send` command executes, the first thing it does is release the CTRL and ALT keys so that the characters get sent properly. After sending all the keys, the command doesn't know whether it can safely push back down CTRL and ALT (to match whether the user is still holding them down). But if less than the specified number of milliseconds have elapsed, it will assume that the user hasn't had a chance to release the keys yet and it will thus push them back down to match their physical state. Otherwise, the modifier keys will not be pushed back down and the user will have to release and press them again to get them to modify the same or another key.

The timeout should be set to a value less than the amount of time that the user typically holds down a hotkey's modifiers before releasing them. Otherwise, the modifiers may be restored to the down position (get stuck down) even when the user isn't physically holding them down.

You can reduce or eliminate the need for this directive with one of the following:

- Install the keyboard hook by adding the line `#InstallKeybdHook` anywhere in the script.
- Use the `SendInput` or `SendPlay` methods rather than the traditional `SendEvent` method.
- When using the traditional `SendEvent` method, reduce `SetKeyDelay` to 0 or -1, which should help because it sends the keystrokes more quickly.

If this directive is unspecified in a script, it behaves as though set to 50.

## Related

[GetKeyState](#)

## Example

```
#HotkeyModifierTimeout 100
```

# #Hotstring

Changes [hotstring](#) options or ending characters.

```
#Hotstring NoMouse
#Hotstring EndChars NewChars
#Hotstring NewOptions
```

## Parameters

### NoMouse

Prevents mouse clicks from resetting the hotstring recognizer as described [here](#). As a side-effect, this also prevents the [mouse hook](#) from being required by hotstrings (though it will still be installed if the script requires it for other purposes, such as mouse hotkeys). The presence of `#Hotstring NoMouse` anywhere in the script affects all hotstrings, not just those physically beneath it.

### EndChars NewChars

Specify the word EndChars followed a single space and then the new ending characters. For example:

```
#Hotstring EndChars -()[\]{}'";"/\, .?!`n `t
```

Since the new ending characters are in effect globally for the entire script -- regardless of where the EndChars directive appears -- there is no need

to specify `EndChars` more than once.

The maximum number of ending characters is 100. Characters beyond this length are ignored.

To make tab an ending character, include ``t` in the list. To make space an ending character, include it between two other characters in the list (or at the beginning if the list contains only one other character, or no other characters).

## NewOptions

Specify new options as described in [Hotstring Options](#). For example:

```
#Hotstring r s k0 c0.
```

Unlike `EndChars` above, the `#Hotstring` directive is positional when used this way. In other words, entire sections of hotstrings can have different default options as in this example:

```
::btw::by the way

#Hotstring r c ; All the below hotstrings
will use "send raw" and will be case
sensitive by default.
::al::airline
::CEO::Chief Executive Officer

#Hotstring c0 ; Make all hotstrings below
this point case insensitive.
```

## Related



# #If

Creates context-sensitive [hotkeys](#) and [hotstrings](#). Such hotkeys perform a different action (or none at all) depending on the result of an expression.

```
#If [Expression]
```

## Parameters

### Expression

Any valid [expression](#).

## Basic Operation

Any valid expression may be used to define the context in which a hotkey should be active. For example:

```
#If WinActive("ahk_class Notepad") or
WinActive(MyWindowTitle)
#Space::MsgBox You pressed Win+Spacebar in
Notepad or %MyWindowTitle%.
```

Like the `#IfWin` directives, `#If` is positional: it affects all hotkeys and hotstrings physically beneath it in the script. `#If` and `#IfWin` are also mutually exclusive; that is, only the most recent `#If` or `#IfWin` will be in effect.

To turn off context sensitivity, specify `#If` or any `#IfWin` directive but omit all

the parameters. For example:

```
#If
```

## General Remarks

When the key combination which forms a hotkey is pressed, the #If expression is evaluated to determine if the hotkey should activate. The system may not respond to keyboard or mouse input until expression evaluation completes or [times out](#). Sending keystrokes or mouse clicks while the expression is being evaluated (such as from a function which it calls) may cause complications and should be avoided.

The expression may also be evaluated whenever the program needs to know whether the hotkey is active. For example, the #If expression for a custom combination like `a & b ::` might be evaluated when the prefix key (`a` in this example) is pressed, to determine whether it should act as a custom modifier key.

For the reasons described above, the expression should be written to complete quickly and without side-effects.

[A\\_ThisHotkey](#) and [A\\_TimeSinceThisHotkey](#) are set based on the hotkey for which the current #If expression is being evaluated.

[A\\_PriorHotkey](#) and [A\\_TimeSincePriorHotkey](#) temporarily contain the previous values of the corresponding "This" variables.

## Related

Most behavioural properties of the `#IfWin` directives also apply to `#If`.

`#IfTimeout` may be used to override the default timeout value.

## Examples

```
; Example 1: Adjust volume by scrolling the mouse wheel over the taskbar.
```

```
#If MouseIsOver("ahk_class Shell_TrayWnd")
WheelUp::Send {Volume_Up}
WheelDown::Send {Volume_Down}

MouseIsOver(WinTitle) {
 MouseGetPos,,, Win
 return WinExist(WinTitle . " ahk_id " . Win)
}
```

```
; Example 2: Simple word-delete shortcuts for all Edit controls.
```

```
#If ActiveControlIsOfClass("Edit")
^BS::Send ^+{Left}{Del}
^Del::Send ^+{Right}{Del}

ActiveControlIsOfClass(Class) {
 FocusedControl := ControlGetFocus("A")
 FocusedControlHwnd :=
ControlGetHwnd(FocusedControl, "A")
 FocusedControlClass := WinGetClass("ahk_id "
FocusedControlHwnd)
 return (FocusedControlClass=Class)
}
```

```
; Example 3: Context-insensitive hotkey.
```

```
#If
Esc::ExitApp
```

```
; Example 4: Dynamic hotkeys. Requires Example 1.
```

```
NumpadAdd::
```

```
Hotkey, If, MouseIsOver("ahk_class Shell_TrayWnd")
```

```
if (doubleup := !doubleup)
```

```
 Hotkey, WheelUp, DoubleUp
```

```
else
```

```
 Hotkey, WheelUp, WheelUp
```

```
return
```

```
DoubleUp:
```

```
Send {Volume_Up 2}
```

```
return
```

# #IfTimeout

Sets the maximum time that may be spent evaluating a single `#if` expression.

**#IfTimeout** Timeout

## Parameters

### Timeout

The timeout value to apply globally, in milliseconds.

## Remarks

A timeout is implemented to prevent long-running expressions from stalling keyboard input processing. If the timeout value is exceeded, the expression continues to evaluate, but the keyboard hook continues as if the expression had already returned false.

If this directive is unspecified in the script, it will behave as though set to 1000.

Note that the system implements its own timeout, defined by the DWORD value *LowLevelHooksTimeout* in the following registry key:

**HKEY\_CURRENT\_USER\Control Panel\Desktop**

If the system timeout value is exceeded, the system may stop calling the script's keyboard hook, thereby preventing hook hotkeys from working until the hook is

re-registered or the script is [reloaded](#). The hook can *usually* be re-registered by [suspending](#) and un-suspending all hotkeys.

If a given hotkey has multiple `#If` variants, the timeout might be applied to each variant independently, making it more likely that the system timeout will be exceeded. This may be changed in a future update.

## Related

[#If](#)

## Example

```
#IfTimeout 10 ; Set the timeout to 10 ms.
```

# #IfWinActive / #IfWinExist

Creates context-sensitive [hotkeys](#) and [hotstrings](#). Such hotkeys perform a different action (or none at all) depending on the type of window that is active or exists.

```
#IfWinActive [WinTitle, WinText]
#IfWinExist [WinTitle, WinText]
#IfWinNotActive [WinTitle, WinText]
#IfWinNotExist [WinTitle, WinText]
#If [Expression]
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

Title matching behaviour is determined by [SetTitleMatchMode](#) as set in the [auto-execute section](#).

As with most other directives, variables are not supported. Although [ahk\\_pid](#) and [ahk\\_id](#) can be used with a hard-coded process or window ID, it is more common for #IfWin to use them indirectly via [GroupAdd](#) or [Hotkey IfWin](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) has been turned on in the auto-execute section (top part of the script).

### ExcludeTitle

### ExcludeText

Although these are **not** supported, they can be used indirectly by specifying `ahk_group MyGroup` for *WinTitle* (where *MyGroup* is a group created via [GroupAdd](#), which supports ExcludeTitle/Text).

## Basic Operation

The `#IfWin` directives make it easy to create context-sensitive [hotkeys](#) and [hotstrings](#). For example:

```
#IfWinActive ahk_class Notepad
#space::MsgBox You pressed Win+Spacebar in
Notepad.
```

The `#IfWin` directives are positional: they affect all hotkeys and hotstrings physically beneath them in the script. They are also mutually exclusive; that is, only the most recent one will be in effect.

To turn off context sensitivity, specify any `#IfWin` directive but omit all of its parameters. For example:

```
#IfWinActive
```

---

When #IfWin is turned off (or never used in a script), all hotkeys and hotstrings are enabled for all windows (unless disabled via Suspend or the Hotkey command).

When a mouse or keyboard hotkey is disabled via #IfWin, it performs its native function; that is, it passes through to the active window as though there is no such hotkey. There is one exception: Joystick hotkeys: although #IfWin works, it never prevents other programs from seeing the press of a button.

#IfWin can also be used to alter the behavior of an ordinary key like Enter or Space. This is useful when a particular window ignores that key or performs some action you find undesirable. For example:

```
#IfWinActive Reminders ahk_class #32770 ; The
"reminders" window in Outlook.
Enter::Send !o ; Have an "Enter" keystroke
open the selected reminder rather than snoozing
it.
#IfWinActive
```

## Variant (Duplicate) Hotkeys

A particular hotkey or hotstring can be defined more than once in the script if each definition has different #IfWin criteria. These are known as hotkey variants. For example:

```
#IfWinActive ahk_class Notepad
^!c::MsgBox You pressed Control+Alt+C in
```

```
Notepad.
#IfWinActive ahk_class WordPadClass
^!c::MsgBox You pressed Control+Alt+C in
WordPad.
#IfWinActive
^!c::MsgBox You pressed Control+Alt+C in a
window other than Notepad/WordPad.
```

If more than one variant is eligible to fire, only the one closest to the top of the script will fire. The exception to this is the global variant (the one with no #IfWin criteria): It always has the lowest precedence; therefore, it will fire only if no other variant is eligible (this exception does not apply to [hotstrings](#)).

When creating duplicate hotkeys, the order of [modifier symbols](#) such as ^!+# does not matter. For example, `^!C` is the same as `!^C`. However, keys must be spelled consistently. For example, *Esc* is not the same as *Escape* for this purpose (though the case does not matter). Also, any hotkey with a [wildcard prefix](#) (\*) is entirely separate from a non-wildcard one; for example, `^*F1` and `*F1` would each have their own set of variants.

To have the same hotkey subroutine executed by more than one variant, the easiest way is to create a stack of identical hotkeys, each with a different #IfWin directive above it. For example:

```
#IfWinActive ahk_class Notepad
#Z::
#IfWinActive ahk_class WordPadClass
#Z::
MsgBox You pressed Win+Z in either Notepad or
WordPad.
```

```
return
```

Alternatively, a [window group](#) can be used via `#IfWinActive ahk_group MyGroup`.

To create hotkey variants dynamically (while the script is running), see "[Hotkey IfWin](#)".

## General Remarks

`#IfWin` also restores prefix keys to their native function when appropriate (a [prefix key](#) is the "a" key in a hotkey such as "a & b"). This occurs whenever there are no enabled hotkeys for a given prefix.

When `Gosub` or `Goto` is used to jump to a hotkey or hotstring label, it jumps to the variant closest to the top of the script.

When a hotkey is currently disabled via `#IfWin`, its key or mouse button will appear with a "#" character in [KeyHistory's](#) "Type" column. This can help debug a script.

Variable references such as `%Var%` are not currently supported. Therefore, percent signs must be [escaped](#) via ``%`` to allow future support for them. Similarly, literal commas must be escaped (via ``,`) to allow additional parameters to be added in the future. If you need to work around this limitation, use [GroupAdd](#) and `ahk_group`.

A label to which the [Hotkey command](#) has assigned a hotkey is not directly

affected by `#IfWin`. Instead, the use of `#IfWin` closest to the bottom of the script (if any) will be in effect for all hotkeys created by the `Hotkey` command (unless "`Hotkey IfWin`" has been used to change that).

`Alt-tab` hotkeys are not affected by `#IfWin`: they are in effect for all windows.

The `Last Found Window` is set by `#IfWinActive/Exist` (though not by `#IfWinNotActive/NotExist`). For example:

```
#IfWinExist ahk_class Notepad
#n::WinActivate ; Activates the window found
by #IfWin.
```

The `escape sequences` ``s` and ``t` may be used if leading or trailing spaces/tabs are needed in one of `#IfWin`'s parameters.

For performance reasons, `#IfWin` does not continuously monitor the activation or existence of the specified windows. Instead, it checks for a matching window only when you type a hotkey or hotstring. If a matching window is not present, your keystroke or mouse click is allowed to pass through to the active window unaltered.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on in the auto-execute section (top part of the script).

## Related

#If *expression*, Hotkey command, Hotkeys, Hotstrings, Suspend, WinActive, WinExist, SetTitleMatchMode, DetectHiddenWindows

## Examples

```
#IfWinActive ahk_class Notepad
^!a::MsgBox You pressed Ctrl-Alt-A while Notepad
is active. ; This hotkey will have no effect if
pressed in other windows (and it will "pass
through").
#c::MsgBox You pressed Win-C while Notepad is
active.
::btw::This replacement text for "btw" will occur
only in Notepad.
#IfWinActive
#c::MsgBox You pressed Win-C in a window other
than Notepad.
```

# #InputLevel

Controls which artificial keyboard and mouse events are ignored by hotkeys and hotstrings.

```
#InputLevel [Level]
```

## Parameters

### Level

An integer between 0 and 100. If omitted, it defaults to 0.

## General Remarks

For an explanation of how SendLevel and #InputLevel are used, see [SendLevel](#).

This directive is positional: it affects all hotkeys and hotstrings between it and the next #InputLevel directive. If not specified by an #InputLevel directive, hotkeys and hotstrings default to level 0.

A hotkey's input level can also be set using the Hotkey command. For example:

```
Hotkey, #z, my_hotkey_sub, I1
```

The input level of a hotkey or non-auto-replace hotstring is also used as the default [send level](#) for any keystrokes or button clicks generated by that hotkey or hotstring. Since a keyboard or mouse [remapping](#) is actually a pair of hotkeys, this allows #InputLevel to be used to allow remappings to trigger other hotkeys.

AutoHotkey versions older than v1.1.06 behave as though #InputLevel 0 and SendLevel 0 are in effect.

## Related

[SendLevel](#), [Hotkeys](#), [Hotstrings](#)

## Examples

```
#InputLevel 1
Numpad0::LButton
#InputLevel 0
; This hotkey can be triggered by both Numpad0 and
LButton:
~LButton::Msgbox Clicked
```

# #MaxHotkeysPerInterval

Along with `#HotkeyInterval`, specifies the rate of `hotkey` activations beyond which a warning dialog will be displayed.

`#MaxHotkeysPerInterval` Value

## Parameters

### Value

The maximum number of hotkeys that can be pressed in the interval specified by `#HotkeyInterval` without triggering a warning dialog.

## Remarks

Care should be taken not to make the above too lenient because if you ever inadvertently introduce an infinite loop of keystrokes (via a `Send` command that accidentally triggers other hotkeys), your computer could become unresponsive due to the rapid flood of keyboard events.

As an oversimplified example, the hotkey `^C::Send ^C` would produce an infinite loop of keystrokes. To avoid this, add the `$` prefix to the hotkey definition (e.g. `^C::Send $^C`) so that the hotkey cannot be triggered by the `Send` command.

If this directive is unspecified in the script, it will behave as though set to 70.

## Related

#HotkeyInterval

## Example

```
#MaxHotkeysPerInterval 200
```

# #MaxThreads

Sets the maximum number of simultaneous [threads](#).

**#MaxThreads** Value

## Parameters

### Value

The maximum total number of [threads](#) that can exist simultaneously. Specifying a number higher than 255 is the same as specifying 255.

## Remarks

This setting is global, meaning that it needs to be specified only once (anywhere in the script) to affect the behavior of the entire script.

Although a value of 1 is allowed, it is not recommended because it would prevent new [hotkeys](#) from launching whenever the script is displaying a [MsgBox](#) or other dialog. It would also prevent [timers](#) from running whenever another [thread](#) is sleeping or waiting.

Up to two of the following types of [threads](#) may be created even when #MaxThreads has been reached: A [hotkey](#), [hotstring](#), [OnClipboardChange](#), or GUI event if the first line of its subroutine is [ExitApp](#), [Pause](#), [Edit](#), [Reload](#), [KeyHistory](#), [ListLines](#), [ListVars](#), or [ListHotkeys](#). Also, the [OnExit](#) callback function will always launch regardless of how many threads exist.

If this setting is lower than `#MaxThreadsPerHotkey`, it effectively overrides that setting.

If this directive is unspecified in the script, it will behave as though set to 10.

## Related

`#MaxThreadsPerHotkey`, `Threads`, `#MaxHotkeysPerInterval`, `#HotkeyInterval`, `ListHotkeys`

## Example

```
#MaxThreads 2
```

# #MaxThreadsBuffer

Causes some or all [hotkeys](#) to buffer rather than ignore keypresses when their [#MaxThreadsPerHotkey](#) limit has been reached.

**#MaxThreadsBuffer** On|Off

## Parameters

### On|Off

**On:** All hotkey subroutines between here and the next `#MaxThreadsBuffer ON` directive will buffer rather than ignore presses of their hotkeys whenever their subroutines are at their [#MaxThreadsPerHotkey](#) limit.

**Off:** This is the default behavior. A hotkey press will be ignored whenever that hotkey is already running its maximum number of threads (usually 1, but this can be changed with [#MaxThreadsPerHotkey](#)).

## Remarks

This directive is rarely used because this type of buffering, which is OFF by default, usually does more harm than good. For example, if you accidentally press a hotkey twice, having this setting ON would cause that hotkey's subroutine to automatically run a second time if its first [thread](#) takes less than 1 second to finish (this type of buffer expires after 1 second, by design). Note that

AutoHotkey buffers hotkeys in several other ways (such as [Thread Interrupt](#) and [Critical](#)). It's just that this particular way can be detrimental, thus it is OFF by default.

The main use for this directive is to increase the responsiveness of the keyboard's auto-repeat feature. For example, when you hold down a hotkey whose `#MaxThreadsPerHotkey` setting is 1 (the default), incoming keypresses are ignored if that hotkey subroutine is already running. Thus, when the subroutine finishes, it must wait for the next auto-repeat keypress to come in, which might take 50ms or more due to being caught in between keystrokes of the auto-repeat cycle. This 50ms delay can be avoided by enabling this directive for any hotkey that needs the best possible response time while it is being auto-repeated.

As with all `#` directives, this one should not be positioned in the script as though it were a command (i.e. it is not necessary to have it contained within a subroutine). Instead, position it immediately before the first hotkey label you wish to have affected by it.

## Related

[#MaxThreads](#), [#MaxThreadsPerHotkey](#), [Critical](#), [Thread \(command\)](#), [Threads](#), [Hotkey](#), [#MaxHotkeysPerInterval](#), [#HotkeyInterval](#), [ListHotkeys](#)

## Example

```
#MaxThreadsBuffer on
```



# #MaxThreadsPerHotkey

Sets the maximum number of simultaneous [threads](#) per [hotkey](#) or [hotstring](#).

**#MaxThreadsPerHotkey** Value

## Parameters

### Value

The maximum number of [threads](#) that can be launched for a given hotkey/hotstring subroutine (limit 255).

## Remarks

This setting is used to control how many "instances" of a given [hotkey](#) or [hotstring](#) subroutine are allowed to exist simultaneously. For example, if a hotkey has a max of 1 and it is pressed again while its subroutine is already running, the press will be ignored. This is helpful to prevent accidental double-presses. However, if you wish these keypresses to be buffered rather than ignored -- perhaps to increase the responsiveness of the keyboard's auto-repeat feature -- use [#MaxThreadsBuffer](#).

Unlike [#MaxThreads](#), this setting is **not** global. Instead, position it before the first hotkey label you wish to have affected by it, which will result in all subsequent hotkeys using that value until another instance of this directive is encountered.

Any [hotkey](#) subroutine whose first line is [ExitApp](#), [Pause](#), [Edit](#), [Reload](#), [KeyHistory](#), [ListLines](#), [ListVars](#), or [ListHotkeys](#) will always run regardless of this setting.

The setting of [#MaxThreads](#) -- if lower than this setting -- takes precedence.

If this directive is unspecified in the script, it will behave as though set to 1.

## Related

[#MaxThreads](#), [#MaxThreadsBuffer](#), [Critical](#), [Threads](#), [Hotkey](#),  
[#MaxHotkeysPerInterval](#), [#HotkeyInterval](#), [ListHotkeys](#)

## Example

```
#MaxThreadsPerHotkey 3
```

# #MenuMaskKey

Changes which key is used to mask Win or Alt keyup events.

```
#MenuMaskKey KeyName
```

## Parameters

### KeyName

A [key name](#) or vkNN sequence which specifies a non-zero virtual keycode. Scan codes are not used.

## Remarks

This setting is global, meaning that it needs to be specified only once (anywhere in the script) to affect the behavior of the entire script.

If a hotkey is implemented using the keyboard hook or mouse hook, the final keypress may be invisible to the active window and the system. For hotkeys which use a Win or Alt modifier key, this could result in the Start menu or active window's menu bar being activated when the modifier key is released. To prevent this from happening, AutoHotkey "masks" the keyup event by sending a keystroke. Prior to Revision 38, this was always a Ctrl keystroke, and was known to cause problems with certain applications.

If this directive is unspecified in the script, it will behave as though set to Ctrl.

## Related

See [this thread](#) for background information.

## Examples

```
#MenuMaskKey vk07 ; vk07 is unassigned.
#UseHook
#Space::
!Space::
 KeyWait LWin
 KeyWait RWin
 KeyWait Alt
 KeyHistory
return
```

# #UseHook

Forces the use of the hook to implement all or some keyboard [hotkeys](#).

```
#UseHook [On | Off]
```

## Parameters

### On|Off

#UseHook without one of the following words after it is equivalent to `#UseHook On`.

**On:** The [keyboard hook](#) will be used to implement all keyboard hotkeys between here and the next `#UseHook OFF` (if any).

**Off:** Hotkeys will be implemented using the default method (RegisterHotkey() if possible; otherwise, the keyboard hook).

## Remarks

Normally, the windows API function RegisterHotkey() is used to implement a keyboard hotkey whenever possible. However, the responsiveness of hotkeys might be better under some conditions if the [keyboard hook](#) is used instead.

Turning this directive ON is equivalent to using the [\\$ prefix](#) in the definition of each affected hotkey.

As with all # directives -- which are processed only once when the script is launched -- `#UseHook` should not be positioned in the script as though it were a command (that is, it is not necessary to have it contained within a subroutine). Instead, position it immediately before the first hotkey label you wish to have affected by it.

By default, hotkeys that use the `keyboard hook` cannot be triggered by means of the `Send` command. Similarly, mouse hotkeys cannot be triggered by commands such as `Click` because all mouse hotkeys use the `mouse hook`. One workaround is to use `Gosub` to jump directly to the hotkey's subroutine. For example: `Gosub #LButton`.

`#InputLevel` and `SendLevel` provide additional control over which hotkeys and hotstrings are triggered by the `Send` command.

If this directive does not appear in the script at all, it will behave as though set to OFF.

## Related

`#InstallKeybdHook`, `#InstallMouseHook`, `ListHotkeys`, `#InputLevel`

## Example

```
#UseHook ; Force the use of the hook for hotkeys after this point.
```

```
#x::MsgBox, This hotkey will be implemented with the hook.
```

```
#y::MsgBox, And this one too.
```



# Hotkey

Creates, modifies, enables, or disables a hotkey while the script is running.

```
Hotkey KeyName [, Label, Options]
```

```
Hotkey "IfWinActive/Exist" [, WinTitle, WinText]
```

```
Hotkey "If" [, Expression]
```

```
Hotkey "If", FunctionObject
```

```
Command Example: Hotkey "!a", "MyHotkeyLabel"
Function Example: Hotkey("!a", "MyHotkeyLabel")
```

## Parameters

### KeyName

Name of the hotkey's activation key, including any [modifier symbols](#). For example, specify `#C` for the Win+C hotkey.

If *KeyName* already exists as a hotkey, that hotkey will be updated with the values of the command's other parameters.

*KeyName* can also be the name of an existing hotkey label (i.e. a double-colon label), which will cause that hotkey to be updated with the values of the command's other parameters.

When specifying an *existing* hotkey, *KeyName* is not case sensitive.

However, the names of keys must be spelled the same as in the existing

hotkey (e.g. Esc is not the same as Escape for this purpose). Also, the order of **modifier symbols** such as `^!+#` does not matter. `GetKeyName` can be used to retrieve the standard spelling of a key name.

When a hotkey is first created -- either by the Hotkey command or a **double-colon label** in the script -- its key name and the ordering of its modifier symbols becomes the permanent name of that hotkey as reflected by `A_ThisHotkey`. This name is shared by all **variants** of the hotkey, and does not change even if the Hotkey command later accesses the hotkey with a different symbol ordering.

If the hotkey variant already exists, its behavior is updated according to whether *KeyName* includes or excludes the **tilde (~) prefix**.

The **use hook (\$) prefix** can be added to existing hotkeys. This prefix affects all variants of the hotkey and cannot be removed.

## Label

The name of the **label** whose contents will be executed (as a new **thread**) when the hotkey is pressed. Both normal labels and **hotkey/hotstring** labels can be used. The trailing colon(s) should not be included. If *Label* is dynamic (e.g. `%VarContainingLabelName%`), `IsLabel(VarContainingLabelName)` may be called beforehand to verify that the label exists.

If not a valid label name, this parameter can be the name of a function, or a single variable reference containing a **function object**. For example,

`Hotkey %funcobj%, On` or `Hotkey % funcobj, On`. Other expressions which return objects are currently unsupported. When the hotkey executes, the function is called without parameters. Hotkeys can also be [defined as functions](#) without the Hotkey command.

This parameter can be left blank if *KeyName* already exists as a hotkey, in which case its label will not be changed. This is useful to change only the hotkey's *Options*.

**If the label or function is specified but the hotkey is disabled from a previous use of this command, the hotkey will remain disabled. To prevent this, include the word ON in *Options*.**

This parameter can also be one of the following special values:

**On:** The hotkey becomes enabled. No action is taken if the hotkey is already On.

**Off:** The hotkey becomes disabled. No action is taken if the hotkey is already Off.

**Toggle:** The hotkey is set to the opposite state (enabled or disabled).

**AltTab** (and others): These are special Alt-Tab hotkey actions that are described [here](#).

**Note:** If there is a label in the script named On, Off, Toggle or AltTab (or any variation recognized by this command), the label is ignored and cannot be used with this command.

## Options

A string of zero or more of the following letters with optional spaces in between. For example: `UseErrorLevel B0`.

**UseErrorLevel:** If the command encounters a problem, this option skips the warning dialog, sets `ErrorLevel` to one of the codes from the table below, then allows the `current thread` to continue.

**On:** Enables the hotkey if it is currently disabled.

**Off:** Disables the hotkey if it is currently enabled. This is typically used to create a hotkey in an initially-disabled state.

**B or B0:** Specify the letter B to buffer the hotkey as described in `#MaxThreadsBuffer`. Specify B0 (B with the number 0) to disable this type of buffering.

**Pn:** Specify the letter P followed by the hotkey's `thread priority`. If the P option is omitted when creating a hotkey, 0 will be used.

**Tn:** Specify the letter T followed by a the number of threads to allow for this hotkey as described in `#MaxThreadsPerHotkey`. For example: `T5`.

**In (InputLevel):** Specify the letter I (or i) followed by the hotkey's `input level`. For example: `I1`.

If any of the option letters are omitted and the hotkey already exists, those options will not be changed. But if the hotkey does not yet exist -- that is,

it is about to be created by this command -- the options will default to those most recently in effect. For example, the instance of `#MaxThreadsBuffer` that occurs closest to the bottom of the script will be used. If `#MaxThreadsBuffer` does not appear in the script, its default setting (OFF in this case) will be used. This behavior also applies to `#IfWin`: the bottommost occurrence applies to newly created hotkeys unless "`Hotkey IfWin`" has executed since the script started.

`IfWinActive`

`IfWinExist`

`IfWinNotActive`

`IfWinNotExist`

`If, Expression`

`If, % FunctionObject`

These sub-commands make all subsequently-created hotkeys context sensitive. See [below](#) for details.

`WinTitle`

`WinText`

Within these parameters, any variable reference such as `%var%` becomes permanent the moment the command finishes. In other words, subsequent changes to the contents of the variable are not seen by existing `IfWin` hotkeys.

Like `#IfWinActive/Exist`, `WinTitle` and `WinText` use the default settings for `SetTitleMatchMode` and `DetectHiddenWindows` as set in the `auto-execute`

section. See [#IfWinActive/Exist](#) for details.

## If, Expression

Associates subsequently-created hotkeys with a given [#If](#) expression. *Expression* must be an expression which has been used with the [#If](#) directive elsewhere in the script. Although this command is unable to create new expressions, it can create new hotkeys using an existing expression. See [#If example 4](#).

**Note:** The Hotkey command uses the string that you pass to it, not the original source code. Commas and deref chars (percent signs) are interpreted *before* the command is called, so may need to be escaped if they are part of the original expression. [Escape sequences](#) are resolved when the script loads, so only the resulting characters are considered; for example, `Hotkey, If, x = "`t"` and `Hotkey, If, % "x = "" A_Tab ""` both correspond to `#If x = "`t"`.

## If, % FunctionObject

Associates subsequently-created hotkeys with a given [function object](#). Such hotkeys will only execute if calling the given function object yields a non-zero number. This is like `Hotkey, If, Expression`, except that each hotkey can have many [variants](#) (one per object). *FunctionObject* must be a single variable (not an expression) containing an object with a *call* method. The function or *call* method can accept one parameter, the [name](#) of the hotkey.

Once passed to the Hotkey command, the object will never be deleted

(but memory will be reclaimed by the OS when the process exits).

The "three-key combination" example below uses this sub-command.

## ErrorLevel

If the first parameter is "If" or begins with "IfWin", an exception is thrown if a parameter is invalid or a memory allocation fails. ErrorLevel is not set in those cases.

ErrorLevel is changed only when the word UseErrorLevel is present in the Options parameter.

| Error | Description                                                                                                                                                                                                                   |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | The <i>Label</i> parameter specifies a nonexistent label name.                                                                                                                                                                |
| 2     | The <i>KeyName</i> parameter specifies one or more keys that are either not recognized or not supported by the current keyboard layout/language.                                                                              |
| 3     | Unsupported prefix key. For example, using the mouse wheel as a prefix in a hotkey such as <code>WheelDown &amp; Enter</code> is not supported.                                                                               |
| 4     | The <i>KeyName</i> parameter is not suitable for use with the <a href="#">AltTab</a> or <a href="#">ShiftAltTab</a> actions. A combination of two keys is required. For example: <code>RControl &amp; RShift::AltTab</code> . |
| 5     | The command attempted to modify a nonexistent hotkey.                                                                                                                                                                         |
| 6     | The command attempted to modify a nonexistent <a href="#">variant</a> of an existing hotkey. To solve this, use <a href="#">Hotkey IfWin</a> to set the criteria to match those of the hotkey to be modified.                 |
| 98    | Creating this hotkey would exceed the 1000-hotkey-per-script limit (however, each hotkey can have an unlimited number of <a href="#">variants</a> , and there is no limit to the number of <a href="#">hotstrings</a> ).      |

99

Out of memory. This is very rare and usually happens only when the operating system has become unstable.

Tip: The UseErrorLevel option can be used to test for the existence of a hotkey variant. For example:

```
Hotkey, ^!p,, UseErrorLevel
if ErrorLevel = 5 or ErrorLevel = 6
 MsgBox The hotkey does not exist or it has
no variant for the current IfWin criteria.
```

## Remarks

The [current IfWin setting](#) determines the [variant](#) of a hotkey upon which the Hotkey command will operate.

If the goal is to disable selected hotkeys or hotstrings automatically based on the type of window that is active, `Hotkey, ^!c, Off` is usually less convenient than using `#IfWinActive/Exist` (or their dynamic counterparts "Hotkey IfWinActive/Exist" [below](#)).

Creating hotkeys via [double-colon labels](#) performs better than using the Hotkey command because the hotkeys can all be enabled as a batch when the script starts (rather than one by one). Therefore, it is best to use this command to create only those hotkeys whose key names are not known until after the script has started running. One such case is when a script's hotkeys for various actions are configurable via an [INI file](#).

A given label can be the target of more than one hotkey. If it is known that a label was called by a hotkey, you can determine which hotkey by checking the built-in variable `A_ThisHotkey`.

If the script is [suspended](#), newly added/enabled hotkeys will also be suspended until the suspension is turned off (unless they are exempt as described in the [Suspend](#) section).

The [keyboard](#) and/or [mouse](#) hooks will be installed or removed if justified by the changes made by this command.

Although the Hotkey command cannot directly enable or disable hotkeys in scripts other than its own, in most cases it can [override](#) them by creating or enabling the same hotkeys. Whether this works depends on a combination of factors: 1) Whether the hotkey to be overridden is a [hook hotkey](#) in the other script (non-hook hotkeys can always be overridden); 2) The fact that the most recently started script's hotkeys generally take precedence over those in other scripts (therefore, if the script intending to override was started most recently, its override should always succeed); 3) Whether the enabling or creating of this hotkey will newly activate the [keyboard](#) or [mouse](#) hook (if so, the override will always succeed).

Once a script has at least one hotkey, it becomes [persistent](#), meaning that [ExitApp](#) rather than `Exit` should be used to terminate it.

## **Remarks About *Hotkey*, If**

The "Hotkey If" commands allow context-sensitive [hotkeys](#) to be created and modified while the script is running (by contrast, the `#If` and `#IfWinActive/Exist` directives are positional and take effect before the script begins executing). For example:

```
Hotkey, IfWinActive, ahk_class Notepad
Hotkey, ^!e, MyLabel ; Creates a hotkey that
works only in Notepad.
```

Using `Hotkey If` puts context sensitivity into effect for all subsequently created or modified [hotkeys](#) in the current [thread](#). In addition, each If sub-command is mutually exclusive; that is, only the most recent one will be in effect.

To turn off context sensitivity (that is, to make subsequently-created hotkeys work in all windows), specify any If sub-command but omit the parameters. For example: `Hotkey, If` or `Hotkey, IfWinActive`.

Before `Hotkey If` is used in a hotkey or hotstring [thread](#), the Hotkey command defaults to the same context as the hotkey or hotstring that launched the thread. In other words, `Hotkey, %A_ThisHotkey%, Off` turns off the current hotkey even if it is context-sensitive. All other threads default to creating or modifying global hotkeys, unless that default is overridden by using `Hotkey If` in the [auto-execute section](#).

When a mouse or keyboard hotkey is disabled via an If sub-command or directive, it performs its native function; that is, it passes through to the active

window as though there is no such hotkey. However, joystick hotkeys always pass through, whether they are disabled or not.

## Variant (Duplicate) Hotkeys

A particular hotkey can be created more than once if each definition has different IfWin criteria. These are known as *hotkey variants*. For example:

```
Hotkey, IfWinActive, ahk_class Notepad
Hotkey, ^!c, MyLabelForNotepad
Hotkey, IfWinActive, ahk_class WordPadClass
Hotkey, ^!c, MyLabelForWordPad
Hotkey, IfWinActive
Hotkey, ^!c, MyLabelForAllOtherWindows
```

If more than one variant of a hotkey is eligible to fire, only the one created earliest will fire. The exception to this is the global variant (the one with no IfWin criteria): It always has the lowest precedence, and thus will fire only if no other variant is eligible.

When creating duplicate hotkeys, the order of [modifier symbols](#) such as ^!+# does not matter. For example, `^!c` is the same as `!^c`. However, keys must be spelled consistently. For example, *Esc* is not the same as *Escape* for this purpose (though the case does not matter). Finally, any hotkey with a [wildcard prefix](#) (\*) is entirely separate from a non-wildcard one; for example, `*F1` and `F1` would each have their own set of variants.

For more information about IfWin hotkeys, see [#IfWin's General Remarks](#).

## Related

Hotkey Symbols, #IfWinActive/Exist, #MaxThreadsBuffer, #MaxThreadsPerHotkey, Suspend, IsLabel, Threads, Thread, Critical, Gosub, Return, Menu, SetTimer

## Examples

```
Hotkey, ^!z, MyLabel
return

MyLabel:
MsgBox You pressed %A_ThisHotkey%.
return
```

**; Other examples:**

Hotkey, RCtrl & RShift, AltTab **; Makes RCtrl & RShift operate like Alt-Tab.**

Hotkey, #c, On **; Re-enables the Win-C hotkey.**

Hotkey, \$+#c, Off **; Disables the Shift-Win-C hotkey.**

Hotkey, ^!a, , T5 **; Changes the hotkey to allow 5 threads.**

```
Hotkey, IfWinActive, ahk_class Notepad
```

```
Hotkey, ^!c, MyLabelForNotepad ; Creates Ctrl-Alt-C as a hotkey that works only in Notepad.
```

**; This GUI allows you to register primitive three-key combination hotkeys:**

```
Gui := GuiCreate()
Gui.Add("Text", "xm", "Prefix key:")
Gui.Add("Edit", "yp x100 w100 vPrefix", "Space")
```



# ListHotkeys

Displays the hotkeys in use by the current script, whether their subroutines are currently running, and whether or not they use the [keyboard](#) or [mouse](#) hook.

## ListHotkeys

This command is equivalent to selecting the View->Hotkeys menu item in the main window.

If a hotkey has been disabled via the [Hotkey](#) command, it will be listed as OFF or PART ("PART" means that only some of the hotkey's [variants](#) are disabled).

If any of a hotkey's variants have a non-zero [#InputLevel](#), the level (or minimum and maximum levels) are displayed.

If any of a hotkey's subroutines are currently running, the total number of threads is displayed for that hotkey.

Finally, the type of hotkey is also displayed, which is one of the following:

reg: The hotkey is implemented via the operating system's RegisterHotkey() function.

reg(no): Same as above except that this hotkey is inactive (due to being unsupported, disabled, or [suspended](#)).

k-hook: The hotkey is implemented via the [keyboard hook](#).

m-hook: The hotkey is implemented via the [mouse hook](#).

2-hooks: The hotkey requires both the hooks mentioned above.

joypoll: The hotkey is implemented by polling the joystick at regular intervals.

## Related

[#InstallKeybdHook](#), [#InstallMouseHook](#), [#UseHook](#), [KeyHistory](#), [ListLines](#), [ListVars](#), [#MaxThreadsPerHotkey](#), [#MaxHotkeysPerInterval](#)

## Example

```
ListHotkeys
```

# #InstallKeybdHook

Forces the unconditional installation of the keyboard hook.

```
#InstallKeybdHook
```

## Remarks

The keyboard hook monitors keystrokes for the purpose of activating [hotstrings](#) and any keyboard [hotkeys](#) not supported by RegisterHotkey (which is a function built into the operating system). It also supports a few other features such as the [Input](#) command.

AutoHotkey does not install the keyboard and mouse hooks unconditionally because together they consume at least 500 KB of memory. Therefore, the keyboard hook is normally installed only when the script contains one of the following: 1) [hotstrings](#); 2) one or more [hotkeys](#) that require the keyboard hook (most do not); 3) [SetCaps/Scroll/Numlock AlwaysOn/AlwaysOff](#); 4) the [Input](#) command, for which the hook is installed upon first actual use.

By contrast, the `#InstallKeybdHook` directive will unconditionally install the keyboard hook, which might be useful to allow [KeyHistory](#) to display the last 20 keystrokes (for debugging purposes), or to avoid the need for [#HotkeyModifierTimeout](#).

You can determine whether a script is using the hook via the [KeyHistory](#) command or menu item. You can determine which hotkeys are using the hook

via the `ListHotkeys` command or menu item.

## Related

[#InstallMouseHook](#), [#UseHook](#), [Hotkey](#), [Input](#), [KeyHistory](#), [Hotstrings](#), [GetKeyState](#), [KeyWait](#)

## Example

```
#InstallKeybdHook
```

# #InstallMouseHook

Forces the unconditional installation of the mouse hook.

```
#InstallMouseHook
```

## Remarks

The mouse hook monitors mouse clicks for the purpose of activating mouse [hotkeys](#) and [facilitating hotstrings](#).

AutoHotkey does not install the keyboard and mouse hooks unconditionally because together they consume at least 500 KB of memory (but if the keyboard hook is installed, installing the mouse hook only requires about 50 KB of additional memory; and vice versa). Therefore, the mouse hook is normally installed only when the script contains one or more mouse [hotkeys](#). It is also installed for [hotstrings](#), but that can be disabled via [#Hotstring NoMouse](#).

By contrast, the `#InstallMouseHook` directive will unconditionally install the mouse hook, which might be useful to allow [KeyHistory](#) to monitor mouse clicks.

You can determine whether a script is using the hook via the [KeyHistory](#) command or menu item. You can determine which hotkeys are using the hook via the [ListHotkeys](#) command or menu item.

## Related

#InstallKeybdHook, #UseHook, Hotkey, KeyHistory, GetKeyState, KeyWait

## Example

```
#InstallMouseHook
```

# #KeyHistory

Sets the maximum number of keyboard and mouse events displayed by the [KeyHistory](#) window. You can set it to 0 to disable key history.

```
#KeyHistory MaxEvents
```

## Parameters

### MaxEvents

The maximum number of keyboard and mouse events displayed by the [KeyHistory](#) window (default 40, limit 500). Specify 0 to disable key history entirely.

## Remarks

Because this setting is put into effect before the script begins running, it is only necessary to specify it once (anywhere in the script).

Because each keystroke or mouse click consists of a down-event and an up-event, [KeyHistory](#) displays only half as many "complete events" as specified here. For example, if the script contains `#KeyHistory 50`, up to 25 keystrokes and mouse clicks will be displayed.

## Related

[KeyHistory](#), [#NoTrayIcon](#)

## Example

```
#KeyHistory 0 ; Disable key history.
#KeyHistory 100 ; Store up to 100 events.
```

# BlockInput

Disables or enables the user's ability to interact with the computer via keyboard and mouse.

**BlockInput** Mode

```
Command Example: BlockInput, "On"
Function Example: BlockInput("Off")
```

## Parameters

### Mode

**Mode 1**: One of the following words:

**On** or 1 ([true](#)): The user is prevented from interacting with the computer (mouse and keyboard input has no effect).

**Off** or 0 ([false](#)): Input is re-enabled.

**Mode 2**: This mode operates independently of the other two. For example, `BlockInput On` will continue to block input until `BlockInput Off` is used, even if one of the below is also in effect.

**Send**: The user's keyboard and mouse input is ignored while a [Send](#) or [SendRaw](#) is in progress (the traditional [SendEvent mode](#) only). This prevents the user's keystrokes from disrupting the flow of simulated

keystrokes. When the *Send* finishes, input is re-enabled (unless still blocked by a previous use of `BlockInput On`).

**Mouse:** The user's keyboard and mouse input is ignored while a `Click`, `MouseMove`, `MouseClick`, or `MouseClickDrag` is in progress (the traditional *SendEvent mode* only). This prevents the user's mouse movements and clicks from disrupting the simulated mouse events. When the mouse command finishes, input is re-enabled (unless still blocked by a previous use of `BlockInput On`).

**SendAndMouse:** A combination of the above two modes.

**Default:** Turns off both the *Send* and the *Mouse* modes, but does not change the current state of input blocking. For example, if `BlockInput On` is currently in effect, using `BlockInput Default` will not turn it off.

**Mode 3:** This mode operates independently of the other two. For example, if `BlockInput On` and `BlockInput MouseMove` are both in effect, mouse movement will be blocked until both are turned off.

**MouseMove:** The mouse cursor will not move in response to the user's physical movement of the mouse (DirectInput applications are a possible exception). When a script first uses this command, the `mouse hook` is installed (if it is not already). The mouse hook will stay installed until the next use of the `Suspend` or `Hotkey` command, at which time it is removed if not required by any hotkeys or hotstrings (see `#Hotstring NoMouse`).

**MouseMoveOff:** Allows the user to move the mouse cursor.

## Remarks

**Note:** `BlockInput On` might have no effect if UAC is enabled and the script has not been run as administrator. For more information, refer to the [FAQ](#).

In preference to `BlockInput`, it is often better to use `SendMode Input` or `SendMode Play` so that keystrokes and mouse clicks become uninterruptible. This is because unlike `BlockInput`, those modes do not discard what the user types during the send; instead, those keystrokes are buffered and sent afterward. Avoiding `BlockInput` also avoids the need to work around sticking keys as described in the next paragraph.

If `BlockInput` becomes active while the user is holding down keys, it might cause those keys to become "stuck down". This can be avoided by waiting for the keys to be released prior to turning `BlockInput` on, as in this example:

```
^!p::
KeyWait Control ; wait for the key to be
released. Use one KeyWait for each of the
hotkey's modifiers.
KeyWait Alt
BlockInput On
; ... send keystrokes and mouse clicks ...
BlockInput Off
return
```

Input blocking is automatically and momentarily disabled whenever an ALT

event is sent (then re-enabled afterward).

When BlockInput is in effect, user input is blocked but AutoHotkey can simulate keystrokes and mouse clicks. However, pressing Ctrl+Alt+Del will re-enable input due to a Windows API feature.

Certain types of [hook hotkeys](#) can still be triggered when BlockInput is on. Examples include `MButton` (mouse hook) and `LWin & Space` (keyboard hook with explicit prefix rather than modifiers "\$#").

Input is automatically re-enabled when the script closes.

## Related

[SendMode](#), [Send](#), [Click](#), [MouseMove](#), [MouseButton](#), [MouseClickDrag](#)

## Example

### Function Syntax

```
BlockInput("on")
Run("notepad")
WinWaitActive("ahk_class Notepad")
Send("{F5}") ; pastes time and date
BlockInput("off")
```

### Command Syntax

```
BlockInput, on
Run, notepad
WinWaitActive, ahk_class Notepad
```

```
Send, {F5} ; pastes time and date
BlockInput, off
```

# Click

Clicks a mouse button at the specified coordinates. It can also hold down a mouse button, turn the mouse wheel, or move the mouse.

Here are examples of common usages (all commas are optional):

```
Click [Arg1, Arg2, ...]
```

```
Command Example: Click "100 100"
Function Example: Click(100,100)
```

## Parameters

|                    |                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------|
| Click (by itself)  | Clicks the left mouse button once at the mouse cursor's current position.                      |
| Click 44, 55       | Clicks the left mouse button once at coordinates 44, 55 (based on <a href="#">CoordMode</a> ). |
| Click right 44, 55 | Same as above but clicks the right mouse button.                                               |
| Click 2            | Clicks the left mouse button twice at the cursor's current position (i.e. double-click).       |
| Click down         | Presses the left mouse button down and holds it.                                               |
| Click up right     | Releases the right mouse button.                                                               |
| Click %x% %y%      | Variables should be enclosed in percent signs.                                                 |

Zero or more of the following items can follow the word *Click*. Separate each item from the next with at least one space, tab, and/or comma. The items can appear in any order except *ClickCount*, which must occur somewhere to the right of the coordinates (if coordinates are present).

**X, Y:** The x/y coordinates to which the mouse cursor is moved prior to clicking. Coordinates are relative to the active window unless `CoordMode` was used to change that. If omitted, the cursor's current position is used.

**Button Name:** Left (default), Right, Middle (or just the first letter of each of these); or the fourth or fifth mouse button (X1 or X2). NOTE: Unlike `MouseClicked`, the left and right buttons behave consistently across all systems, even if the user has swapped the buttons via the system's control panel.

**Mouse Wheel:** Specify `WheelUp` or `WU` to turn the wheel upward (away from you); specify `WheelDown` or `WD` to turn the wheel downward (toward you). `WheelLeft` (or `WL`) or `WheelRight` (or `WR`) may also be specified (but they have no effect on older operating systems older than Windows Vista). For *ClickCount* (below), specify the number of notches to turn the wheel. However, some applications do not obey a *ClickCount* higher than 1 for the mouse wheel. For them, use a `Loop` such as the following:

```
Loop 5
 Click WheelUp
```

**ClickCount:** The number of times to click the mouse (examples: `Click 2`, `Click 100, 200, 2`). If omitted, the button is clicked once. If coordinates

are specified, *ClickCount* must appear after them. Specify zero (0) to move the mouse without clicking (for example: `Click 100, 200, 0`).

**Down or Up:** These words are normally omitted, in which case each click consists of a down-event followed by an up-event. Otherwise, specify *Down* (or the letter *D*) to press the mouse button down without releasing it. Later, use the word *Up* (or the letter *U*) to release the mouse button.

**Relative:** The word *Rel* or *Relative* treats the specified X and Y coordinates as offsets from the current mouse position. In other words, the cursor will be moved from its current position by X pixels to the right (left if negative) and Y pixels down (up if negative).

## Remarks

*Click* is generally preferred over *MouseClicked* because it automatically compensates if the user has swapped the left and right mouse buttons via the system's control panel.

*Click* uses the sending method set by *SendMode*. To override this mode for a particular click, use a specific Send command as in this example: `SendEvent {Click, 100, 200}`.

To perform a shift-click or control-click, the `Send {Click}` method is generally easiest. For example:

```
Send +{Click 100, 200} ; Shift+LeftClick
Send ^{Click 100, 200, right} ;
```

## Control+RightClick

Unlike `Send`, `Click` does not automatically release the modifier keys (Control, Alt, Shift, and Win). For example, if the Control key is currently down, `Click` would produce a control-click but `Send {Click}` would produce a normal click.

The `SendPlay mode` is able to successfully generate mouse events in a broader variety of games than the other modes. In addition, some applications and games may have trouble tracking the mouse if it moves too quickly, in which case `SetDefaultMouseSpeed` can be used to reduce the speed (but only in `SendEvent mode`).

The `BlockInput` command can be used to prevent any physical mouse activity by the user from disrupting the simulated mouse events produced by the mouse commands. However, this is generally not needed for the `SendInput` and `SendPlay` modes because they automatically postpone the user's physical mouse activity until afterward.

There is an automatic delay after every click-down and click-up of the mouse (except for `SendInput mode` and for turning the mouse wheel). Use `SetMouseDelay` to change the length of the delay.

## Related

`Send {Click}`, `SendMode`, `CoordMode`, `SetDefaultMouseSpeed`, `SetMouseDelay`, `MouseClicked`, `MouseClickedDrag`, `MouseMove`, `ControlClick`,

## BlockInput

### Examples

`Click` ; Click left mouse button at mouse cursor's current position.

`Click 100, 200` ; Click left mouse button at specified coordinates.

`Click 100, 200, 0` ; Move the mouse without clicking.

`Click 100, 200 right` ; Click the right mouse button.

`Click 2` ; Perform a double-click.

`Click down` ; Presses down the left mouse button and holds it.

`Click up right` ; Releases the right mouse button.

# ControlClick

Sends a mouse button or mouse wheel event to a control.

```
ControlClick [Control-or-Pos, WinTitle, WinText,
WhichButton, ClickCount, Options, ExcludeTitle,
ExcludeText]
```

```
Command example: ControlClick "Edit1", "MyGui
ahk_class AutoHotkeyGUI",,,,, "NA Pos x10 y30"
Function example: ControlClick("Edit1", "MyGui
ahk_class AutoHotkeyGUI",,,,, "NA Pos x10 y30")
```

## Parameters

### Control-or-Pos (optional)

If this parameter is blank, the target window's topmost control will be clicked (or the target window itself if it has no controls). Otherwise, one of the two modes below will be used.

Mode 1 (Position): Specify the X and Y coordinates relative to the upper left corner of the target window's [client area](#). The X coordinate must precede the Y coordinate and there must be at least one space or tab between them. For example: X55 Y33. If there is a control at the specified coordinates, it will be sent the click-event at those exact coordinates. If there is no control, the target window itself will be sent the event (which might have no effect depending on the nature of the

window). Note: In this mode, the X and Y option letters of the *Options* parameter are ignored.

Mode 2 (ClassNN or Text): Specify either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by [SetTitleMatchMode](#).

By default, mode 2 takes precedence over mode 1. For example, in the unlikely event that there is a control whose text or ClassNN has the format "Xnnn Ynnn", it would be acted upon by Mode 2. To override this and use mode 1 unconditionally, specify the word Pos in *Options* as in the following example: `ControlClick, x255 y152, WinTitle,,,, Pos`.

To operate upon a control's HWND (window handle), leave this parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle (optional)

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText (optional)

If present, this parameter must be a substring from a single text element

of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### WhichButton (optional)

The button to click: LEFT, RIGHT, MIDDLE (or just the first letter of each of these). If omitted or blank, the LEFT button will be used.

X1 (XButton1, the 4th mouse button) and X2 (XButton2, the 5th mouse button) are also supported.

WheelUp (or WU) and WheelDown (or WD) are also supported. In this case, *ClickCount* is the number of notches to turn the wheel.

Windows Vista or later: WheelLeft (or WL) and WheelRight (or WR) are also supported (they have no effect on older operating systems). In this case, *ClickCount* is the number of notches to turn the wheel.

### ClickCount (optional)

The number of clicks to send. If omitted or blank, 1 click is sent.

### Options (optional)

A series of zero or more of the following option letters. For example: d x50 y25

**NA:** May improve reliability. See [reliability](#) below.

**D:** Press the mouse button down but do not release it (i.e. generate a down-event). If both the **D** and **U** options are absent, a complete click

(down and up) will be sent.

**U**: Release the mouse button (i.e. generate an up-event). This option should not be present if the **D** option is already present (and vice versa).

**Pos**: Specify the word Pos anywhere in *Options* to unconditionally use the X/Y positioning mode as described in the *Control-or-Pos* parameter above.

**Xn**: Specify for **n** the X position to click at, relative to the control's upper left corner. If unspecified, the click will occur at the horizontal-center of the control.

**Yn**: Specify for **n** the Y position to click at, relative to the control's upper left corner. If unspecified, the click will occur at the vertical-center of the control.

Use decimal (not hexadecimal) numbers for the **X** and **Y** options.

### ExcludeTitle (optional)

Windows whose titles include this value will not be considered.

### ExcludeText (optional)

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Reliability

To improve reliability -- especially during times when the user is physically moving the mouse during the ControlClick -- one or both of the following may help:

1) Use `SetControlDelay -1` prior to ControlClick. This avoids holding the mouse button down during the click, which in turn reduces interference from the user's physical movement of the mouse.

2) Specify the string NA anywhere in the sixth parameter (*Options*) as shown below:

```
SetControlDelay -1
ControlClick, Toolbar321, WinTitle,,,, NA
```

**NA** avoids marking the target window as active and avoids merging its input processing with that of the script, which may prevent physical movement of the mouse from interfering (but usually only when the target window is not active). However, this method might not work for all types of windows and controls.

## Remarks

Not all applications obey a *ClickCount* higher than 1 for turning the mouse wheel. For those applications, use a Loop to turn the wheel more than one notch as in this example, which turns it 5 notches:

```
Loop, 5
```

```
ControlClick, Control, WinTitle, WinText,
WheelUp
```

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[SetControlDelay](#), [Control](#) functions, [ControlGetText](#), [ControlMove](#), [ControlGetPos](#), [ControlFocus](#), [ControlSetText](#), [ControlSend](#), [Click](#)

## Examples

```
ControlClick, OK, Some Window Title ; Clicks the
OK button
ControlClick, x55 y77, WinTitle ; Clicks at a set
of coordinates. Note the lack of a comma between X
and Y.

; The following method may improve reliability and
reduce side effects:
SetControlDelay -1
ControlClick, Toolbar321, WinTitle,,,, NA x192 y10
; Clicks in NA mode at coordinates that are
relative to a named control.
```

# ControlSend / ControlSendRaw

Sends simulated keystrokes to a window or control.

```
ControlSend Keys [, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: ControlSend "Test", "Edit1",
"MyGui ahk_class AutoHotkeyGUI"
```

```
Function Example: ControlSend("Test", "Edit1",
"MyGui ahk_class AutoHotkeyGUI")
```

## Parameters

### Keys

The sequence of keys to send (see the [Send](#) command for details). To send a literal comma, [escape](#) it (`,`). The rate at which characters are sent is determined by [SetKeyDelay](#).

Unlike the [Send](#) command, mouse clicks cannot be sent by [ControlSend](#). Use [ControlClick](#) for that.

### Control

Can be either ClassNN (the classname and instance number of the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank or omitted, the target

window's topmost control will be used. If this parameter is `ahk_parent`, the keystrokes will be sent directly to the target window instead of one of its controls (see [Automating Winamp](#) for an example).

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

`ControlSendRaw` sends the keystrokes in the *Keys* parameter without translating `{Enter}` to an ENTER keystroke, `{^C}` to Control-C, etc. For details, see [Raw mode](#). It is also valid to use `{Raw}` with `ControlSend`.

If the *Control* parameter is omitted, this command will attempt to send directly to the target window by sending to its topmost control (which is often the correct one) or the window itself if there are no controls. This is useful if a window does not appear to have any controls at all, or just for the convenience of not having to worry about which control to send to.

By default, modifier keystrokes (Control, Alt, Shift, and Win) are sent as they normally would be by the `Send` command. This allows command prompt and other console windows to properly detect uppercase letters, control characters, etc. It may also improve reliability in other ways.

However, in some cases these modifier events may interfere with the active window, especially if the user is actively typing during a `ControlSend` or if the Alt key is being sent (since Alt activates the active window's menu bar). This can be avoided by explicitly sending modifier up and down events as in this example:

```
ControlSend, Edit1, {Alt down}f{Alt up},
Untitled - Notepad
```

The method above also allows the sending of modifier keystrokes (Control/Alt/Shift/Win) while the workstation is locked (protected by logon prompt).

`BlockInput` should be avoided when using `ControlSend` against a console window such as command prompt. This is because it might prevent capitalization and modifier keys such as Control from working properly.

The value of `SetKeyDelay` determines the speed at which keys are sent. If the target window does not receive the keystrokes reliably, try increasing the press duration via the second parameter of `SetKeyDelay` as in these examples:

```
SetKeyDelay, 10, 10
SetKeyDelay, 0, 10
SetKeyDelay, -1, 0
```

If the target control is an Edit control (or something similar), the following are usually more reliable and faster than `ControlSend`:

```
ControlEditPaste("This text will be inserted at
the caret position.", ControlName, WinTitle)
```

```
ControlSetText("This text will entirely replace
any current text.", ControlName, WinTitle)
```

`ControlSend` is generally not capable of manipulating a window's menu bar. To work around this, use `MenuSelect`. If that is not possible due to the nature of the menu bar, you could try to discover the message that corresponds to the desired

menu item by following the [SendMessage Tutorial](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[SetKeyDelay](#), [Escape sequences \(e.g. `%\)](#), [Control functions](#), [ControlGetText](#), [ControlMove](#), [ControlGetPos](#), [ControlClick](#), [ControlSetText](#), [ControlFocus](#), [Send](#), [Automating Winamp](#)

## Examples

### Function Syntax

```
Run("notepad", "", "MIN", PID) ; Run
Notepad minimized.
WinWait("ahk_pid " PID) ; Wait for
it to appear.
; Send the text to the inactive Notepad edit
control.
; The third parameter is omitted so the last found
window is used.
ControlSend("Edit1", "This is a line of text in
the notepad window.{Enter}")
ControlSendRaw("Edit1", "Notice that {Enter} is
not sent as an Enter keystroke with
ControlSendRaw.")

Msgbox("Press OK to activate the window to see the
result.")
WinActivate("ahk_pid " PID) ; Show the
result.
```

```

SetTitleMatchMode(2)
Run(A_ComSpec, "", "", PID) ;
Run command prompt.
WinWait("ahk_pid " PID) ;
Wait for it to appear.
ControlSend("", "ipconfig{Enter}", "cmd.exe") ;
Send directly to the command prompt window.

```

## Command Syntax

```

Run, notepad,, MIN, PID ; Run Notepad
minimized.
WinWait, ahk_pid %PID ; Wait for it to
appear.
; Send the text to the inactive Notepad edit
control.
; The third parameter is omitted so the last found
window is used.
ControlSend, Edit1, This is a line of text in the
notepad window.{Enter}
ControlSendRaw, Edit1, Notice that {Enter} is not
sent as an Enter keystroke with ControlSendRaw.

Msgbox, Press OK to activate the window to see the
result.
WinActivate, ahk_pid %PID ;Show the result.

```

```

SetTitleMatchMode, 2
Run, %A_ComSpec,,, PID ;
Run command prompt.
WinWait, ahk_pid %PID ;
Wait for it to appear.
ControlSend,, ipconfig{Enter}, cmd.exe ;
Send directly to the command prompt window.

```



# CoordMode

Sets coordinate mode for various commands to be relative to either the active window or the screen.

**CoordMode** Command [, Mode]

```
Command Example: CoordMode "ToolTip", "Screen"
Function Example: CoordMode("ToolTip", "Screen")
```

## Parameters

### Param1

**ToolTip:** Affects [ToolTip](#).

**Pixel:** Affects [PixelGetColor](#), [PixelSearch](#), and [ImageSearch](#).

**Mouse:** Affects [MouseGetPos](#), [Click](#), and [MouseMove/Click/Drag](#).

**Caret:** Affects the built-in variables [A\\_CaretX](#) and [A\\_CaretY](#).

**Menu:** Affects the [Menu Show](#) command when coordinates are specified for it.

### Param2

If *Param2* is omitted, it defaults to [Screen](#).

**Screen or 2:** Coordinates are relative to the desktop (entire screen).

**Window or Relative or 1:** Coordinates are relative to the active window.

**Client or 0:** Coordinates are relative to the active window's client area, which excludes the window's title bar, menu (if it has a standard one) and borders. Client coordinates are less dependent on OS version and theme.

## Remarks

If this command is not used, all commands except those documented otherwise (e.g. [WinMove](#) and [InputBox](#)) use coordinates that are relative to the active window's client area.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

The built-in [A\\_CoordMode](#) variables contain the current settings.

## Related

[Click](#), [MouseMove](#), [MouseClicked](#), [MouseClickedDrag](#), [MouseGetPos](#), [PixelGetColor](#), [PixelSearch](#), [ToolTip](#), [Menu](#)

## Example

```
CoordMode, ToolTip, Screen ; Place ToolTips at
absolute screen coordinates:
CoordMode, ToolTip ; Same effect as the above
```

because "screen" is the default.

# GetKeyName/VK/SC()

Retrieves the name/text, virtual key code or scan code of a key.

```
String := GetKeyName(Key)
Number := GetKeyVK(Key)
Number := GetKeySC(Key)
```

## Parameters

### Key

A VK or SC code, such as "vkA2" or "sc01D", a combination of both, or a [key name](#). For example, both `GetKeyName("vk1B")` and `GetKeyName("Esc")` return "Escape", while `GetKeyVK("Esc")` returns 27. Note that VK and SC codes must be in hexadecimal. To convert a decimal number to the appropriate format, use `Format("vk{:x}", vk_code)` or `Format("sc{:x}", sc_code)`.

## Return Value

These functions return a name, virtual key code or scan code of *Key*.

## Related

[GetKeyState](#), [Key List](#), [Format](#)

## Examples

```
; Show information for a specific key.
key := "lwin" ; Any key can be used here.

name := GetKeyName(key)
vk := GetKeyVK(key)
sc := GetKeySC(key)

MsgBox, %
Format("Name: `t{}``nVK: `t{:X}``nSC: `t{:X}`", name,
vk, sc)
```

# GetKeyState

Checks if a keyboard key or mouse/joystick button is down or up. Also retrieves joystick status.

```
OutputVar := GetKeyState(KeyName [, Mode])
```

```
Function Example: State := GetKeyState("Space",
"P")
```

## Parameters

### KeyName

This can be just about any single character from the keyboard or one of the key names from the [key list](#), such as a mouse/joystick button.

Examples: B, 5, LWin, RControl, Alt, Enter, Escape, LButton, MButton, Joy1.

Alternatively, an explicit virtual key code such as vkFF may be specified. This is useful in the rare case where a key has no name. The virtual key code of such a key can be determined by following the steps at the bottom of the [key list page](#).

**Known limitation:** This function cannot differentiate between two keys which share the same virtual key code, such as Left and NumpadLeft.

### Mode

This parameter is ignored when retrieving joystick status.

If omitted, the mode will default to that which retrieves the logical state of the key. This is the state that the OS and the active window believe the key to be in, but is not necessarily the same as the physical state.

Alternatively, one of these letters may be specified:

**P:** Retrieve the physical state (i.e. whether the user is physically holding it down). The physical state of a key or mouse button will usually be the same as the logical state unless the keyboard and/or mouse hooks are installed, in which case it will accurately reflect whether or not the user is physically holding down the key or button (as long as it was pressed down while the script was running). You can determine if your script is using the hooks via the [KeyHistory](#) command or menu item. You can force the hooks to be installed by adding the [#InstallKeybdHook](#) and/or [#InstallMouseHook](#) directives to the script.

**T:** Retrieve the toggle state (only meaningful for keys that can be toggled such as Capslock, Numlock, Scrolllock, and Insert). A retrieved value of 1 (true) means the key is "on", while 0 (false) means it's "off".

## Return Value

For keyboard and mouse keys, the return value is 1 if the key is down (or toggled on) or 0 if it is up (or toggled off).

If *KeyName* is invalid or the state of the key could not be determined, an empty

string is returned.

The items below apply only to joysticks:

1) For a joystick axis such as JoyX, *OutputVar* will be set to a floating point number between 0 and 100 to indicate the joystick's position as a percentage of that axis's range of motion. This [test script](#) can be used to analyze your joystick(s).

2) When *KeyName* is JoyPOV, the retrieved value will be between 0 and 35900.

The following approximate POV values are used by many joysticks:

-1: no angle to report

0: forward POV

9000 (i.e. 90 degrees): right POV

27000 (i.e. 270 degrees): left POV

18000 (i.e. 180 degrees): backward POV

## Remarks

To wait for a key or mouse/joystick button to achieve a new state, it is usually easier to use [KeyWait](#) instead of a GetKeyState loop.

Systems with unusual keyboard drivers might be slow to update the state of their keys, especially the toggle-state of keys like Capslock. A script that checks the state of such a key immediately after it changed may use [Sleep](#) beforehand to give the system time to update the key state.

For examples of using GetKeyState with a joystick, see the [joystick remapping](#)

page and the Joystick-To-Mouse script.

## Related

[KeyWait](#), [Key List](#), [Joystick remapping](#), [KeyHistory](#), [#InstallKeybdHook](#),  
[#InstallMouseHook](#)

## Examples

```
; Basic examples:
GetKeyState, state, RButton ; Right mouse button.
GetKeyState, state, Joy2 ; The second button of
the first joystick.
```

```
GetKeyState, state, Shift
if state
 MsgBox At least one Shift key is down.
else
 MsgBox Neither Shift key is down.
```

```
state := GetKeyState("Capslock", "T") ; True if
CapsLock is ON, false otherwise.
```

```
; Remapping example (this is only for illustration
because it would be easier to use
; the built-in remapping feature):
; In the following hotkey, the mouse button is
kept held down while NumpadAdd is
; down, which effectively transforms NumpadAdd
into a mouse button. This method
; can also be used to repeat an action while the
user is holding down a key or button:
```

```
*NumpadAdd::
MouseClicked, left,,, 1, 0, D ; Hold down the left
```

mouse button.

```
Loop
```

```
{
```

```
 Sleep, 10
```

```
 GetKeyState, state, NumpadAdd, P
```

```
 if !state ; The key has been released, so
```

```
break out of the loop.
```

```
 break
```

```
 ; ... insert here any other actions you want
```

```
repeated.
```

```
}
```

```
MouseClick, left,,, 1, 0, U ; Release the mouse
```

```
button.
```

```
return
```

```
; Example: Make joystick button behavior depend on
joystick axis position.
```

```
joy2::
```

```
GetKeyState, joyx, JoyX
```

```
if joyx > 75
```

```
 MsgBox Action #1 (button pressed while
joystick was pushed to the right).
```

```
else if joyx < 25
```

```
 MsgBox Action #2 (button pressed while
joystick was pushed to the left).
```

```
else
```

```
 MsgBox Action #3 (button pressed while
joystick was centered horizontally).
```

```
return
```

```
; See the joystick remapping page and the
Joystick-To-Mouse script for other examples.
```

# KeyHistory

Displays script info and a history of the most recent keystrokes and mouse clicks.

## KeyHistory

```
Command Example: KeyHistory
Function Example: KeyHistory()
```

## Remarks

This command is equivalent to selecting the "View->Key history" menu item in the main window.

To disable key history, specify the following line anywhere in the script:

```
#KeyHistory 0
```

`#KeyHistory` can also be used to change the maximum number of events that will be displayed.

This feature is intended to help [debug scripts and hotkeys](#). It can also be used to detect the scan code of a non-standard keyboard key using the steps described at the bottom of the [key list](#) page (knowing the scan code allows such a key to be made into a hotkey).

The virtual key (VK) of the wheel events (WheelDown, WheelUp, WheelLeft, and WheelRight) are placeholder values that do not have any meaning outside of AutoHotkey. Also, the scan code for wheel events is actually the number of notches by which the wheel was turned (typically 1).

If the script does not have the [keyboard hook](#) installed, the KeyHistory window will display only the keyboard events generated by the script itself (i.e. not the user's). If the script does not have the [mouse hook](#) installed, mouse button events will not be shown. You can find out if your script uses either hook via "View->Key History" in the script's main window (accessible via "Open" in the tray icon). You can force the hooks to be installed by adding either or both of the following lines to the script:

```
#InstallKeybdHook
#InstallMouseHook
```

## Related

[#KeyHistory](#), [#InstallKeybdHook](#), [#InstallMouseHook](#), [ListHotkeys](#), [ListLines](#), [ListVars](#), [GetKeyState](#), [KeyWait](#), [A\\_PriorKey](#)

## Examples

```
KeyHistory ; Display the history info in a window.
```

# KeyWait

Waits for a key or mouse/joystick button to be released or pressed down.

```
KeyWait KeyName [, Options]
```

```
Command Example: KeyWait "Enter", "L1"
Function Example: KeyWait("Enter", "L1")
```

## Parameters

### KeyName

This can be just about any single character from the keyboard or one of the key names from the [key list](#), such as a mouse/joystick button. Joystick attributes other than buttons are not supported.

An explicit virtual key code such as `VKFF` may also be specified. This is useful in the rare case where a key has no name and produces no visible character when pressed. Its virtual key code can be determined by following the steps at the bottom fo the [key list page](#).

### Options

If this parameter is blank, the command will wait indefinitely for the specified key or mouse/joystick button to be physically released by the user. However, if the [keyboard hook](#) is not installed and *KeyName* is a keyboard key released artificially by means such as the [Send](#) command,

the key will be seen as having been physically released. The same is true for mouse buttons when the [mouse hook](#) is not installed.

Options: A string of one or more of the following letters (in any order, with optional spaces in between):

**D:** Wait for the key to be pushed down.

**L:** Check the logical state of the key, which is the state that the OS and the active window believe the key to be in (not necessarily the same as the physical state). This option is ignored for joystick buttons.

**T:** Timeout (e.g. `T3`). The number of seconds to wait before timing out and setting [ErrorLevel](#) to 1. If the key or button achieves the specified state, the command will not wait for the timeout to expire. Instead, it will immediately set [ErrorLevel](#) to 0 and the script will continue executing.

The timeout value can be a floating point number such as 2.5, but it should not be a hexadecimal value such as 0x03.

## ErrorLevel

[ErrorLevel](#) is set to 1 if the command timed out or 0 otherwise.

## Remarks

The physical state of a key or mouse button will usually be the same as the logical state unless the keyboard and/or mouse hooks are installed, in which case

it will accurately reflect whether or not the user is physically holding down the key. You can determine if your script is using the hooks via the [KeyHistory](#) command or menu item. You can force either or both of the hooks to be installed by adding the [#InstallKeybdHook](#) and [#InstallMouseHook](#) directives to the script.

While the command is in a waiting state, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

To wait for two or more keys to be released, use `KeyWait` consecutively. For example:

```
KeyWait Control ; Wait for both Control and
Alt to be released.
KeyWait Alt
```

To wait for any one key among a set of keys to be pressed down, see the examples section of the [Input](#) command.

## Related

[GetKeyState](#), [Key List](#), [Input](#), [KeyHistory](#), [#InstallKeybdHook](#),  
[#InstallMouseHook](#), [ClipWait](#), [WinWait](#)

## Examples

```
; Example #1: Basic usage:
KeyWait, a ; Wait for the A key to be released.
KeyWait, LButton, D ; Wait for the left mouse
```

button to be pressed down.

```
KeyWait, Joy1, D T3 ; Wait up to 3 seconds for
the first joystick button to be pressed down.
```

```
KeyWait, LAlt, L ; Wait for the left Alt key to
be logically released.
```

; Example #2: A simple hotkey:

```
~Capslock::
```

```
KeyWait, Capslock ; Wait for user to physically
release it.
```

```
MsgBox You pressed and released the Capslock key.
return
```

; Example #3: Remapping a key or mouse button

(this is only for illustration because it

; would be easier to use the [built-in remapping  
feature](#)):

; The left mouse button is kept held down while  
NumpadAdd is down, which effectively

; transforms NumpadAdd into the left mouse button.

```
*NumpadAdd::
```

```
MouseClicked, left,,, 1, 0, D ; Hold down the left
mouse button.
```

```
KeyWait, NumpadAdd ; Wait for the key to be
released.
```

```
MouseClicked, left,,, 1, 0, U ; Release the mouse
button.
```

```
return
```

```
; Example #4: Detects when a key has been double-pressed (similar to double-click).
; KeyWait is used to stop the keyboard's auto-repeat feature from creating an unwanted
; double-press when you hold down the RControl key to modify another key. It does this by
; keeping the hotkey's thread running, which blocks the auto-repeats by relying upon
; #MaxThreadsPerHotkey being at its default setting of 1.
; Note: There is a more elaborate script to distinguish between single, double, and
; triple-presses at the bottom of the SetTimer page.
```

```
~RControl::
if (A_PriorHotkey <> "-RControl" or
A_TimeSincePriorHotkey > 400)
{
 ; Too much time between presses, so this isn't a double-press.
 KeyWait, RControl
 return
}
MsgBox You double-pressed the right control key.
return
```

# Input

Waits for the user to type a string.

```
Text := Input([Options, EndKeys, MatchList])
InputEnd()
```

```
Command Example: Input "L1 A"
Function Example: Input("L1 A")
```

## Parameters

### Text (return value)

The text entered by the user (by default, artificial input is also captured).

The text consists of characters produced by keystrokes according to the active window's keyboard layout/language. Consequently, keystrokes that do not produce characters (such as PageUp and Escape) are not stored (though they can be recognized via the *EndKeys* parameter below).

Whitespace characters such as TAB (``t`) are stored literally. ENTER is stored as linefeed (``n`).

### Options

A string of zero or more of the following letters (in any order, with

optional spaces in between):

**A:** Append input to variable contents. If the option is used, variable content will not be deleted and characters will be appended to variable.

**B:** Backspace is ignored. Normally, pressing backspace during an Input will remove the most recently pressed character from the end of the string. Note: If the input text is visible (such as in an editor) and the arrow keys or other means are used to navigate within it, backspace will still remove the last character rather than the one behind the caret (insertion point).

**C:** Case sensitive. Normally, *MatchList* is not case sensitive.

**I:** Ignore input generated by any AutoHotkey script, such as the `SendEvent` command. However, the `SendInput` and `SendPlay` methods are always ignored, regardless of this setting.

**L:** Length limit (e.g. `L5`). The maximum allowed length of the input. When the text reaches this length, the Input will be terminated and `ErrorLevel` will be set to the word Max unless the text matches one of the *MatchList* phrases, in which case `ErrorLevel` is set to the word Match. If unspecified, the length limit is 16383, which is also the absolute maximum.

**M:** Modified keystrokes such as Control-A through Control-Z are recognized and transcribed if they correspond to real ASCII characters. Consider this example, which recognizes Control-C:

```
CtrlC := Chr(3) ; Store the character for
Ctrl-C in the CtrlC var.
OutputVar := Input("L1 M")
if (OutputVar = CtrlC)
 MsgBox "You pressed Control-C."
ExitApp
```

Note: The characters Ctrl-A through Ctrl-Z correspond to [Chr\(1\)](#) through [Chr\(26\)](#). Also, the **M** option might cause some keyboard shortcuts such as Ctrl-LeftArrow to misbehave while an Input is in progress.

**T**: Timeout (e.g. [T3](#)). The number of seconds to wait before terminating the Input and setting [ErrorLevel](#) to the word Timeout. If the Input times out, *OutputVar* will be set to whatever text the user had time to enter. This value can be a floating point number such as 2.5.

**V**: Visible. Normally, the user's input is blocked (hidden from the system). Use this option to have the user's keystrokes sent to the active window.

**\***: Wildcard (find anywhere). Normally, what the user types must exactly match one of the *MatchList* phrases for a match to occur. Use this option to find a match more often by searching the entire length of the input text.

**E**: Handle single-character end keys by character code instead of by keycode. This provides more consistent results if the active window's keyboard layout is different to the script's keyboard layout. It also prevents key combinations which don't actually produce the given end

characters from ending input; for example, if `@` is an end key, on the US layout Shift+2 will trigger it but Ctrl+Shift+2 will not (if the E option is used). If the C option is also used, the end character is case-sensitive.

## EndKeys

A list of zero or more keys, any one of which terminates the Input when pressed (the *EndKey* itself is not written to *OutputVar*). When an Input is terminated this way, *ErrorLevel* is set to the word EndKey followed by a colon and the name of the *EndKey*. Examples: `EndKey: .`, `EndKey: Escape`.

The *EndKey* list uses a format similar to the *Send* command. For example, specifying `{Enter}. {Esc}` would cause either ENTER, period (.), or ESCAPE to terminate the Input. To use the braces themselves as end keys, specify `{{}}` and/or `{}}`.

To use Control, Alt, or Shift as end-keys, specify the left and/or right version of the key, not the neutral version. For example, specify `{LControl}{RControl}` rather than `{Control}`.

Although modified keys such as Control-C (^c) are not supported, certain characters that require the shift key to be held down -- namely punctuation marks such as `?!:@&{ }` -- are supported. Other characters are supported with the `E` option described above.

An explicit virtual key code such as `{VKFF}` may also be specified. This is useful in the rare case where a key has no name and produces no visible

character when pressed. Its virtual key code can be determined by following the steps at the bottom fo the [key list page](#).

## MatchList

A comma-separated list of key phrases, any of which will cause the Input to be terminated (in which case [ErrorLevel](#) will be set to the word Match). The entirety of what the user types must exactly match one of the phrases for a match to occur (unless the [\\* option](#) is present). In addition, **any spaces or tabs around the delimiting commas are significant**, meaning that they are part of the match string. For example, if *MatchList* is "ABC , XYZ ", the user must type a space after ABC or before XYZ to cause a match.

Two consecutive commas results in a single literal comma. For example, the following would produce a single literal comma at the end of string: "string1,,string2". Similarly, the following list contains only a single item with a literal comma inside it: "single,,item".

Because the items in *MatchList* are not treated as individual parameters, the list can be contained entirely within a variable. In fact, all or part of it must be contained in a variable if its length exceeds 16383 since that is the maximum length of any script line. For example, *MatchList* might be the [expression](#) `List1 "," List2 "," List3` -- where each of the variables contains a large sub-list of match phrases.

## ErrorLevel

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NewInput            | The Input was terminated by another <a href="#">thread</a> calling the Input function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| End                 | The Input was terminated by another <a href="#">thread</a> calling the <a href="#">InputEnd</a> function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Max                 | The Input reached the maximum allowed length and it does not match any of the items in <i>MatchList</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Timeout             | The Input timed out.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Match               | The Input matches one of the items in <i>MatchList</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| EndKey: <i>name</i> | <p>One of the <i>EndKeys</i> was pressed to terminate the Input. In this case, <a href="#">ErrorLevel</a> contains the word EndKey followed by a colon and the name of the terminating key without braces, e.g. "EndKey:Enter", "EndKey:Escape", etc.</p> <p>Note that <i>name</i> is the "normalized" name of the key regardless of how it was written in <i>EndKeys</i>. For example, {Esc} and {vk1B} both produce "ErrorLevel:Escape". <a href="#">GetKeyName</a> can be used to retrieve the normalized name.</p> <p>If the <a href="#">E option</a> was used, <i>name</i> is the actual character which was typed (if applicable). Otherwise, the key name is determined according to the script's active keyboard layout. Single-character key names without the E option are usually lower-case, since the shift state is ignored.</p> |

## Remarks

A `_Input` variable can be used to retrieve current text of Input in multi-threading environment or using `SetTimer`.

If this command is used while an Input is already in progress in another [thread](#), that Input will be terminated and its [ErrorLevel](#) will be set to the word

NewInput. After that (if parameters were given), the new Input will commence.

While an Input is in progress, new [threads](#) such as [custom menu items](#) and [timed subroutines](#) can still be created. Similarly, keyboard [hotkeys](#) are still in effect if the Input is [visible](#). If the Input is not visible, only [hook hotkeys](#) can be triggered.

When a script first uses this command, the [keyboard hook](#) is installed (if it is not already). The keyboard hook will stay installed until the next use of the [Suspend](#) or [Hotkey](#) command, at which time it is removed if not required by any hotkeys or hotstrings.

If you use multiple languages or keyboard layouts, Input uses the keyboard layout of the active window rather than the script's (regardless of whether the Input is [visible](#)).

Although not as flexible, [hotstrings](#) are generally easier to use than the Input command.

## InputEnd

Terminates any Input in progress in another [thread](#).

### InputEnd

Returns true if an Input was in progress, otherwise false.

## Related

KeyWait, Hotstrings, InputBox, #InstallKeybdHook, Threads

## Examples

```
; Wait for the user to press any key. Keys that
produce no visible character -- such as
; the modifier keys, function keys, and arrow keys
-- are listed as end keys so that they
; will be detected too.
```

```
SingleChar := Input("L1", "{LControl}{RControl}
{LAlt}{RAlt}{LShift}{RShift}{LWin}{RWin}{AppsKey}
{F1}{F2}{F3}{F4}{F5}{F6}{F7}{F8}{F9}{F10}{F11}
{F12}{Left}{Right}{Up}{Down}{Home}{End}{PgUp}
{PgDn}{Del}{Ins}{BS}{Capslock}{Numlock}
{PrintScreen}{Pause}")
if InStr(ErrorLevel, "EndKey:")
 SingleKey := SubStr(ErrorLevel, 8)
else
 SingleKey := SingleChar
MsgBox SingleKey
```

```
; This is a working hotkey example. Since the
hotkey has the tilde (~)
; prefix, its own keystroke will pass through to
the active window.
; Thus, if you type [btw (or one of the other
match
; phrases) in any editor, the script will
automatically perform an
; action of your choice (such as replacing the
typed text):
```

```
~[::
UserInput := Input("V T5 L4 C", "{enter}.{esc}
{tab}", "btw,otoh,fl,ahk,ca")
if (ErrorLevel = "Max")
{
 MsgBox "You entered " UserInput ", which is
the maximum length of text."
 return
}
if (ErrorLevel = "Timeout")
{
 MsgBox "You entered " UserInput " at which
time the input timed out."
 return
}
if (ErrorLevel = "NewInput")
 return
If InStr(ErrorLevel, "EndKey:")
{
 MsgBox "You entered " UserInput " and
terminated the input with " ErrorLevel "."
 return
}
; Otherwise, a match was found.
if (UserInput = "btw")
 Send "{backspace 4}by the way"
else if (UserInput = "otoh")
 Send "{backspace 5}on the other hand"
else if (UserInput = "fl")
 Send "{backspace 3}Florida"
else if (UserInput = "ca")
 Send "{backspace 3}California"
else if (UserInput = "ahk")
 Run "https://autohotkey.com"
return
```

# MouseClicked

Clicks or holds down a mouse button, or turns the mouse wheel. NOTE: The [Click command](#) is generally more flexible and easier to use.

```
MouseClicked [WhichButton , X, Y, ClickCount, Speed,
"D|U", "R"]
```

## Parameters

### WhichButton

The button to click: Left (default), Right, Middle (or just the first letter of each of these); or the fourth or fifth mouse button (X1 or X2). For example: `MouseClicked, X1`. This parameter may be omitted, in which case it defaults to Left.

Rotate the mouse wheel: Specify WheelUp or WU to turn the wheel upward (away from you); specify WheelDown or WD to turn the wheel downward (toward you). Specify WheelLeft (or WL) or WheelRight (or WR) to push the wheel left or right, respectively (but these have no effect on operating systems older than Windows Vista). *ClickCount* is the number of notches to turn the wheel.

To compensate automatically for cases where the user has swapped the left and right mouse buttons via the system's control panel, use the [Click command](#) instead.

## X, Y

The x/y coordinates to which the mouse cursor is moved prior to clicking. Coordinates are relative to the active window unless [CoordMode](#) was used to change that. If omitted, the cursor's current position is used.

## ClickCount

The number of times to click the mouse. If omitted, the button is clicked once.

## Speed

The speed to move the mouse in the range 0 (fastest) to 100 (slowest). Note: a speed of 0 will move the mouse instantly. If omitted, the default speed (as set by [SetDefaultMouseSpeed](#) or 2 otherwise) will be used.

*Speed* is ignored for [SendInput/Play](#) modes; they move the mouse instantaneously (though [SetMouseDelay](#) has a mode that applies to [SendPlay](#)). To visually move the mouse more slowly -- such as a script that performs a demonstration for an audience -- use [SendEvent \[Click 100, 200\]](#) or [SendMode Event](#) (optionally in conjunction with [BlockInput](#)).

## D|U

If this parameter is omitted, each click will consist of a "down" event followed by an "up" event. Alternatively:

D = Press the mouse button down but do not release it (i.e. generate a down-event).

U = Release the mouse button (i.e. generate an up-event).

## R

If this parameter is the letter R, the X and Y coordinates will be treated as offsets from the current mouse position. In other words, the cursor will be moved from its current position by X pixels to the right (left if negative) and Y pixels down (up if negative).

## Remarks

This command uses the sending method set by [SendMode](#).

The [Click command](#) is recommended over `MouseClicked` because:

1. It automatically compensates when the left and right mouse buttons are swapped via the control panel.
2. It is generally easier to use.

To perform a shift-click or control-click, use the [Send](#) command before and after the operation as shown in these examples:

### **; Example #1:**

```
Send, {Control down}
MouseClicked, left, 55, 233
Send, {Control up}
```

### **; Example #2:**

```
Send, {Shift down}
MouseClicked, left, 55, 233
Send, {Shift up}
```

---

The [SendPlay mode](#) is able to successfully generate mouse events in a broader variety of games than the other modes. In addition, some applications and games may have trouble tracking the mouse if it moves too quickly. The *speed* parameter or [SetDefaultMouseSpeed](#) can be used to reduce the speed (in the default [SendEvent mode](#) only).

Some applications do not obey a *ClickCount* higher than 1 for the mouse wheel. For them, use a [Loop](#) such as the following:

```
Loop, 5
 MouseClick, WheelUp
```

The [BlockInput](#) command can be used to prevent any physical mouse activity by the user from disrupting the simulated mouse events produced by the mouse commands. However, this is generally not needed for the [SendInput/Play](#) modes because they automatically postpone the user's physical mouse activity until afterward.

There is an automatic delay after every click-down and click-up of the mouse (except for [SendInput mode](#) and for turning the mouse wheel). Use [SetMouseDelay](#) to change the length of the delay.

## Related

[CoordMode](#), [SendMode](#), [SetDefaultMouseSpeed](#), [SetMouseDelay](#), [Click](#), [MouseClicked](#), [MouseMove](#), [ControlClick](#), [BlockInput](#)

## Examples

```
; Double click at the current mouse pos:
```

```
MouseClicked, left
```

```
MouseClicked, left
```

```
; Same as above:
```

```
MouseClicked, left, , , 2
```

```
; Move to specified coordinates then right-click
once:
```

```
MouseClicked, right, 200, 300
```

```
; Here are two hotkeys that simulate the turning
of the mouse wheel:
```

```
#up::MouseClicked, WheelUp, , , 2 ; Turn it by two
notches.
```

```
#down::MouseClicked, WheelDown, , , 2
```

# MouseClickedDrag

Clicks and holds the specified mouse button, moves the mouse to the destination coordinates, then releases the button.

```
MouseClickedDrag WhichButton, X1, Y1, X2, Y2, Speed, "R"
```

## Parameters

### WhichButton

The button to click: Left, Right, Middle (or just the first letter of each of these). Specify X1 for the fourth button and X2 for the fifth. For example:

```
MouseClickedDrag, X1, ...
```

To compensate automatically for cases where the user has swapped the left and right mouse buttons via the system's control panel, use the [Click command](#) instead.

### X1, Y1

The x/y coordinates of the drag's starting position (the mouse will be moved to these coordinates right before the drag is started). Coordinates are relative to the active window unless [CoordMode](#) was used to change that. If omitted, the mouse's current position is used.

### X2, Y2

The x/y coordinates to drag the mouse to (that is, while the button is held down). Coordinates are relative to the active window unless [CoordMode](#) was used to change that.

## Speed

The speed to move the mouse in the range 0 (fastest) to 100 (slowest). Note: a speed of 0 will move the mouse instantly. If omitted, the default speed (as set by [SetDefaultMouseSpeed](#) or 2 otherwise) will be used.

*Speed* is ignored for [SendInput/Play](#) modes; they move the mouse instantaneously (though [SetMouseDelay](#) has a mode that applies to [SendPlay](#)). To visually move the mouse more slowly -- such as a script that performs a demonstration for an audience -- use [SendEvent](#) [\[Click 100, 200\]](#) or [SendMode Event](#) (optionally in conjunction with [BlockInput](#)).

## R

If this parameter is the letter R, the X1 and Y1 coordinates will be treated as offsets from the current mouse position. In other words, the cursor will be moved from its current position by X1 pixels to the right (left if negative) and Y1 pixels down (up if negative).

Similarly, the X2 and Y2 coordinates will be treated as offsets from the X1 and Y1 coordinates. For example, the following would first move the cursor down and to the right by 5 pixels from its starting position, and then drag it from that position down and to the right by 10 pixels:

```
MouseClickDrag, Left, 5, 5, 10, 10, , R
```

## Remarks

This command uses the sending method set by [SendMode](#).

Dragging can also be done via the various Send commands, which is more flexible because the mode can be specified via the command name. For example:

```
SendEvent {Click 6, 52, down}{click 45, 52, up}
```

Another advantage of the method above is that unlike [MouseClickedDrag](#), it automatically compensates when the user has swapped the left and right mouse buttons via the system's control panel.

The [SendPlay mode](#) is able to successfully generate mouse events in a broader variety of games than the other modes. However, dragging via [SendPlay](#) might not work in RichEdit controls (and possibly others) such as those of WordPad and Metapad.

Some applications and games may have trouble tracking the mouse if it moves too quickly. The *speed* parameter or [SetDefaultMouseSpeed](#) can be used to reduce the speed (in the default [SendEvent mode](#) only).

The [BlockInput](#) command can be used to prevent any physical mouse activity by the user from disrupting the simulated mouse events produced by the mouse commands. However, this is generally not needed for the [SendInput/Play](#) modes because they automatically postpone the user's physical mouse activity until afterward.

There is an automatic delay after every click-down and click-up of the mouse (except for [SendInput mode](#)). This delay also occurs after the movement of the mouse during the drag operation. Use [SetMouseDelay](#) to change the length of the delay.

## Related

[CoordMode](#), [SendMode](#), [SetDefaultMouseSpeed](#), [SetMouseDelay](#), [Click](#), [MouseClicked](#), [MouseGetPos](#), [MouseMove](#), [BlockInput](#)

## Example

```
MouseClickedDrag, left, 0, 200, 600, 400
```

**; The following example opens MS Paint and draws a little house:**

```
Run, mspaint.exe
WinWaitActive, ahk_class MSPaintApp,, 2
if ErrorLevel
 return
MouseClickedDrag, L, 150, 250, 150, 150
MouseClickedDrag, L, 150, 150, 200, 100
MouseClickedDrag, L, 200, 100, 250, 150
MouseClickedDrag, L, 250, 150, 150, 150
MouseClickedDrag, L, 150, 150, 250, 250
MouseClickedDrag, L, 250, 250, 250, 150
MouseClickedDrag, L, 250, 150, 150, 250
MouseClickedDrag, L, 150, 250, 250, 250
```

# MouseGetPos

Retrieves the current position of the mouse cursor, and optionally which window and control it is hovering over.

```
MouseGetPos [OutputVarX, OutputVarY, OutputVarWin,
OutputVarControl, 1|2|3]
```

```
Command Example: MouseGetPos, x, y, win, ctrl
Function Example: MouseGetPos(x, y, win, ctrl)
```

## Parameters

### OutputVarX/Y

The names of the variables in which to store the X and Y coordinates. The retrieved coordinates are relative to the active window unless [CoordMode](#) was used to change to screen coordinates.

### OutputVarWin

This optional parameter is the name of the variable in which to store the [unique ID number](#) of the window under the mouse cursor. If the window cannot be determined, this variable will be made blank.

The window does not have to be active to be detected. Hidden windows cannot be detected.

### OutputVarControl

This optional parameter is the name of the variable in which to store the name (ClassNN) of the control under the mouse cursor. If the control cannot be determined, this variable will be made blank.

The names of controls should always match those shown by the Window Spy. However, unlike Window Spy, the window under the mouse cursor does not have to be active for a control to be detected.

1|2|3

If omitted, it defaults to 0. Otherwise, specify one of the following digits:

**1:** Uses a simpler method to determine *OutputVarControl*. This method correctly retrieves the active/topmost child window of an Multiple Document Interface (MDI) application such as SysEdit or TextPad. However, it is less accurate for other purposes such as detecting controls inside a GroupBox control.

**2:** Stores the [control's HWND](#) in *OutputVarControl* rather than the control's ClassNN.

**3:** A combination of 1 and 2 above.

## Remarks

Any of the output variables may be omitted if the corresponding information is not needed.

## Related

CoordMode, WinGet, SetDefaultMouseSpeed, Click

## Example

```
MouseGetPos, xpos, ypos
Msgbox, The cursor is at X%xpos% Y%ypos%.
```

```
; This example allows you to move the mouse around
to see
; the title of the window currently under the
cursor:
```

```
SetTimer, WatchCursor, 100
return
```

```
WatchCursor:
MouseGetPos, , , id, control
WinGetTitle, title, ahk_id %id%
WinGetClass, class, ahk_id %id%
ToolTip, ahk_id %id%`nahk_class
%class%`n%title%`nControl: %control%
return
```

# MouseMove

Moves the mouse cursor.

**MouseMove** X, Y [, Speed, R]

|                 |                 |                     |
|-----------------|-----------------|---------------------|
| <b>Command</b>  | <b>Example:</b> | MouseMove 100, 200  |
| <b>Function</b> | <b>Example:</b> | MouseMove(100, 200) |

## Parameters

### X, Y

The x/y coordinates to move the mouse to. Coordinates are relative to the active window unless [CoordMode](#) was used to change that.

### Speed

The speed to move the mouse in the range 0 (fastest) to 100 (slowest). Note: a speed of 0 will move the mouse instantly. If omitted, the default speed (as set by [SetDefaultMouseSpeed](#) or 2 otherwise) will be used.

*Speed* is ignored for [SendInput/Play](#) modes; they move the mouse instantaneously (though [SetMouseDelay](#) has a mode that applies to [SendPlay](#)). To visually move the mouse more slowly -- such as a script that performs a demonstration for an audience -- use [SendEvent {Click 100, 200}](#) or [SendMode Event](#) (optionally in conjunction with [BlockInput](#)).

## R

If this parameter is the letter R, the X and Y coordinates will be treated as offsets from the current mouse position. In other words, the cursor will be moved from its current position by X pixels to the right (left if negative) and Y pixels down (up if negative).

## Remarks

This command uses the sending method set by [SendMode](#).

The [SendPlay mode](#) is able to successfully generate mouse events in a broader variety of games than the other modes. In addition, some applications and games may have trouble tracking the mouse if it moves too quickly. The *speed* parameter or [SetDefaultMouseSpeed](#) can be used to reduce the speed (in the default [SendEvent mode](#) only).

The [BlockInput](#) command can be used to prevent any physical mouse activity by the user from disrupting the simulated mouse events produced by the mouse commands. However, this is generally not needed for the [SendInput/Play](#) modes because they automatically postpone the user's physical mouse activity until afterward.

There is an automatic delay after every movement of the mouse (except for [SendInput mode](#)). Use [SetMouseDelay](#) to change the length of the delay.

The following is an alternate way to move the mouse cursor that may work better in certain multi-monitor configurations:

```
DllCall("SetCursorPos", int, 100, int, 400) ;
The first number is the X-coordinate and the
second is the Y (relative to the screen).
```

On a related note, the mouse cursor can be temporarily hidden via the [hide-cursor example](#).

## Related

[CoordMode](#), [SendMode](#), [SetDefaultMouseSpeed](#), [SetMouseDelay](#), [Click](#),  
[MouseClicked](#), [MouseClickedDrag](#), [MouseGetPos](#), [BlockInput](#)

## Example

```
; Move the mouse to a new position:
MouseMove, 200, 100
```

```
; Move the mouse slowly (speed 50 vs. 2) by 20
pixels to the right and 30 pixels down
; from its current location:
MouseMove, 20, 30, 50, R
```

# Send / SendRaw / SendInput / SendPlay / SendEvent: Send Keys & Clicks

Sends simulated keystrokes and mouse clicks to the [active](#) window.

**Send** Keys

**SendRaw** Keys

**SendInput** Keys

**SendPlay** Keys

**SendEvent** Keys

```
Command Example: Send "Hello{Space}World{!}"
Function Example: Send("Hello{Space}World{!}")
```

## Parameters

### Keys

The sequence of keys to send. As with other commands, the comma in front of the first parameter is optional.

**Raw mode** - *SendRaw* or `{Raw}`: The characters `^+!#{}` are interpreted literally rather than translating `{Enter}` to an ENTER keystroke, `^C` to Control-C, etc. To use raw mode with *SendInput*, *SendPlay*, or *SendEvent*, write `{Raw}` as the first item in the string; for example: `SendInput {Raw}abc`.

*Raw mode* does not affect the interpretation of [escape sequences](#), [variable](#)

references and expressions. For example, `SendRaw, ``100`%` sends the string `100%`. When using `ControlSend`, it is also necessary to escape literal commas (`,`).

**Normal mode:** When not in raw mode, the following symbols have special meaning: `!+^#{ }`

The modifiers `+^#` affect only the very next key. To send the corresponding modifier key on its own, enclose the key name in braces. To just press (hold down) or release the key, follow the key name with the word "down" or "up" as shown below.

| Symbol | Key                                                                                                                                                                                                  | Press                      | Release                | Examples                                                                                 |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|------------------------|------------------------------------------------------------------------------------------|
| !      | {Alt}                                                                                                                                                                                                | {Alt down}                 | {Alt up}               | <code>Send !a</code> presses ALT+a                                                       |
| +      | {Shift}                                                                                                                                                                                              | {Shift down}               | {Shift up}             | <code>Send +abC</code> sends the text "AbC"<br><code>Send !+a</code> presses ALT+SHIFT+a |
| ^      | {Ctrl}                                                                                                                                                                                               | {Ctrl down}                | {Ctrl up}              | <code>Send ^{Home}</code> presses CONTROL+HOME                                           |
| #      | {LWin}<br>{RWin}                                                                                                                                                                                     | {LWin down}<br>{RWin down} | {LWin up}<br>{RWin up} | <code>Send #e</code> holds down the Windows key and then presses the letter "e"          |
| Symbol | Meaning                                                                                                                                                                                              |                            |                        |                                                                                          |
| { }    | Braces are used to enclose key names and other options, and to send special characters literally. For example, <code>{Tab}</code> is the Tab key and <code>{!}</code> is a literal exclamation mark. |                            |                        |                                                                                          |

**Note:** As capital letters are produced by sending the SHIFT key, **A** produces a different effect in some programs than **a**. For example, **!A** presses ALT+SHIFT+A and **!a** presses ALT+a. If in doubt, use lowercase.

**SendInput and SendPlay:** SendInput and SendPlay use the same syntax as SendEvent but are generally faster and more reliable. In addition, they buffer any physical keyboard or mouse activity during the send, which prevents the user's keystrokes from being interspersed with those being sent. By default, *Send* is synonymous *SendPlay*; but it can be made a synonym for SendEvent or SendPlay via [SendMode](#). For more details about each mode, see [SendInput and SendPlay](#) below.

**SendEvent:** SendEvent sends keystrokes using the Windows `keybd_event` function (search MSDN for details). The rate at which keystrokes are sent is determined by [SetKeyDelay](#). [SendMode](#) can be used to make Send synonymous with SendEvent or SendPlay.

**Key Names:** The following table lists the special keys that can be sent (each key name must be enclosed in braces):

| Key Name     | Resulting Keystroke                               |
|--------------|---------------------------------------------------|
| {F1} - {F24} | Function keys. For example: {F12} is the F12 key. |
| {!}          | !                                                 |
| {#}          | #                                                 |
| {+}          | +                                                 |
| {^}          | ^                                                 |

|                     |                                                                                                                                                                        |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {{}                 | {                                                                                                                                                                      |
| }}                  | }                                                                                                                                                                      |
| {Enter}             | ENTER key on the main keyboard                                                                                                                                         |
| {Escape} or {Esc}   | ESCAPE                                                                                                                                                                 |
| {Space}             | SPACE (this is only needed for spaces that appear either at the beginning or the end of the string to be sent -- ones in the middle can be literal spaces)             |
| {Tab}               | TAB                                                                                                                                                                    |
| {Backspace} or {BS} | Backspace                                                                                                                                                              |
| {Delete} or {Del}   | Delete                                                                                                                                                                 |
| {Insert} or {Ins}   | Insert                                                                                                                                                                 |
| {Up}                | Up-arrow key on main keyboard                                                                                                                                          |
| {Down}              | Down-arrow down key on main keyboard                                                                                                                                   |
| {Left}              | Left-arrow key on main keyboard                                                                                                                                        |
| {Right}             | Right-arrow key on main keyboard                                                                                                                                       |
| {Home}              | Home key on main keyboard                                                                                                                                              |
| {End}               | End key on main keyboard                                                                                                                                               |
| {PgUp}              | Page-up key on main keyboard                                                                                                                                           |
| {PgDn}              | Page-down key on main keyboard                                                                                                                                         |
|                     |                                                                                                                                                                        |
| {CapsLock}          | CapsLock (using <a href="#">SetCapsLockState</a> is more reliable on Win 2k/XP). Sending {CapsLock} might require <a href="#">SetStoreCapslockMode Off</a> beforehand. |
| {ScrollLock}        | ScrollLock (see also: <a href="#">SetScrollLockState</a> )                                                                                                             |
| {NumLock}           | NumLock (see also: <a href="#">SetNumLockState</a> )                                                                                                                   |
|                     |                                                                                                                                                                        |

|                               |                                                                                                                                   |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| {Control} or {Ctrl}           | CONTROL (technical info: sends the neutral virtual key but the left scan code)                                                    |
| {LControl} or {LCtrl}         | Left CONTROL key (technical info: sends the left virtual key rather than the neutral one)                                         |
| {RControl} or {RCtrl}         | Right CONTROL key                                                                                                                 |
| {Control Down} or {Ctrl Down} | Holds the CONTROL key down until {Ctrl Up} is sent. To hold down the left or right key instead, use {RCtrl Down} and {RCtrl Up}.  |
|                               |                                                                                                                                   |
| {Alt}                         | ALT (technical info: sends the neutral virtual key but the left scan code)                                                        |
| {LAlt}                        | Left ALT key (technical info: sends the left virtual key rather than the neutral one)                                             |
| {RAlt}                        | Right ALT key (or AltGr, depending on keyboard layout)                                                                            |
| {Alt Down}                    | Holds the ALT key down until {Alt Up} is sent. To hold down the left or right key instead, use {RAlt Down} and {RAlt Up}.         |
|                               |                                                                                                                                   |
| {Shift}                       | SHIFT (technical info: sends the neutral virtual key but the left scan code)                                                      |
| {LShift}                      | Left SHIFT key (technical info: sends the left virtual key rather than the neutral one)                                           |
| {RShift}                      | Right SHIFT key                                                                                                                   |
| {Shift Down}                  | Holds the SHIFT key down until {Shift Up} is sent. To hold down the left or right key instead, use {RShift Down} and {RShift Up}. |
|                               |                                                                                                                                   |
| {LWin}                        | Left Windows key                                                                                                                  |

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {RWin}      | Right Windows key                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| {LWin Down} | Holds the left Windows key down until {LWin Up} is sent                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| {RWin Down} | Holds the right Windows key down until {RWin Up} is sent                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| {AppsKey}   | Windows App key (invokes the right-click or context menu)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| {Sleep}     | Computer SLEEP key.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| {ASC nnnnn} | <p>Sends an ALT+nnnnn keypad combination, which can be used to generate special characters that don't exist on the keyboard. To generate ASCII characters, specify a number between 1 and 255. To generate ANSI characters (standard in most languages), specify a number between 128 and 255, but precede it with a leading zero, e.g. {Asc 0133}.</p> <p>Unicode characters may be generated by specifying a number between 256 and 65535 (without a leading zero). However, this is not supported by all applications. For alternatives, see the section below.</p> |
| {U+nnnn}    | <p>Sends a Unicode character where <i>nnnn</i> is the hexadecimal value of the character excluding the 0x prefix. This typically isn't needed, because Send and ControlSend automatically support Unicode text.</p> <p>If the character doesn't map to a virtual keycode, <a href="#">SendInput</a> or <a href="#">WM_CHAR</a> is used to send the character and the current Send mode has no effect.</p>                                                                                                                                                              |
|             | Sends a keystroke that has virtual key XX and scan                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                                             |                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>{vkXX}<br/> {scYYY}<br/> {vkXXscYYY}</p> | <p>code YYY. For example: <code>Send {vkFFsc159}</code>. If the sc or vk portion is omitted, the most appropriate value is sent in its place.</p> <p>The values for XX and YYY are hexadecimal and can usually be determined from the main window's <a href="#">View-&gt;Key history</a> menu item. See also: <a href="#">Special Keys</a></p> |
|                                             |                                                                                                                                                                                                                                                                                                                                                |
| <p>{Numpad0} -<br/> {Numpad9}</p>           | <p>Numpad digit keys (as seen when Numlock is ON). For example: {Numpad5} is the digit 5.</p>                                                                                                                                                                                                                                                  |
| <p>{NumpadDot}</p>                          | <p>Numpad Period (as seen when Numlock is ON).</p>                                                                                                                                                                                                                                                                                             |
| <p>{NumpadEnter}</p>                        | <p>Enter key on keypad</p>                                                                                                                                                                                                                                                                                                                     |
| <p>{NumpadMult}</p>                         | <p>Numpad Multiply</p>                                                                                                                                                                                                                                                                                                                         |
| <p>{NumpadDiv}</p>                          | <p>Numpad Divide</p>                                                                                                                                                                                                                                                                                                                           |
| <p>{NumpadAdd}</p>                          | <p>Numpad Add</p>                                                                                                                                                                                                                                                                                                                              |
| <p>{NumpadSub}</p>                          | <p>Numpad Subtract</p>                                                                                                                                                                                                                                                                                                                         |
|                                             |                                                                                                                                                                                                                                                                                                                                                |
| <p>{NumpadDel}</p>                          | <p>Delete key on keypad (this key and the following Numpad keys are used when Numlock is OFF)</p>                                                                                                                                                                                                                                              |
| <p>{NumpadIns}</p>                          | <p>Insert key on keypad</p>                                                                                                                                                                                                                                                                                                                    |
| <p>{NumpadClear}</p>                        | <p>Clear key on keypad (usually the '5' key when Numlock is OFF).</p>                                                                                                                                                                                                                                                                          |
| <p>{NumpadUp}</p>                           | <p>Up-arrow key on keypad</p>                                                                                                                                                                                                                                                                                                                  |
| <p>{NumpadDown}</p>                         | <p>Down-arrow key on keypad</p>                                                                                                                                                                                                                                                                                                                |
| <p>{NumpadLeft}</p>                         | <p>Left-arrow key on keypad</p>                                                                                                                                                                                                                                                                                                                |
| <p>{NumpadRight}</p>                        | <p>Right-arrow key on keypad</p>                                                                                                                                                                                                                                                                                                               |
| <p>{NumpadHome}</p>                         | <p>Home key on keypad</p>                                                                                                                                                                                                                                                                                                                      |

|                     |                                                                                          |
|---------------------|------------------------------------------------------------------------------------------|
| {NumpadEnd}         | End key on keypad                                                                        |
| {NumpadPgUp}        | Page-up key on keypad                                                                    |
| {NumpadPgDn}        | Page-down key on keypad                                                                  |
|                     |                                                                                          |
| {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/unmute the master volume. Usually equivalent to <code>SoundSet, +1, , mute</code> . |
| {Volume_Down}       | Reduce the master volume. Usually equivalent to <code>SoundSet -5</code> .               |
| {Volume_Up}         | Increase the master volume. Usually equivalent to <code>SoundSet +5</code> .             |
| {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                                                                         |
|                     |                                                                                          |

|                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {PrintScreen}                                                                                                                      | Print Screen                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| {CtrlBreak}                                                                                                                        | Ctrl+break                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| {Pause}                                                                                                                            | Pause                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>{Click [Options]}</b>                                                                                                           | <p>Sends a mouse click using the same options available in the <a href="#">Click</a> command. For example, <code>{Click}</code> would click the left mouse button once at the mouse cursor's current position, and <code>{Click 100, 200}</code> would click at coordinates 100, 200 (based on <a href="#">CoordMode</a>). To move the mouse without clicking, specify 0 after the coordinates; for example: <code>{Click 100, 200, 0}</code>. The delay between mouse clicks is determined by <a href="#">SetMouseDelay</a> (not <a href="#">SetKeyDelay</a>).</p> |
| {WheelDown},<br>{WheelUp},<br>{WheelLeft},<br>{WheelRight},<br>{LButton},<br>{RButton},<br>{MButton},<br>{XButton1},<br>{XButton2} | <p>Sends a mouse button event at the cursor's current position (to have control over position and other options, use <a href="#">Click</a> above). The delay between mouse clicks is determined by <a href="#">SetMouseDelay</a>. WheelLeft/Right have no effect on operating systems older than Windows Vista.</p>                                                                                                                                                                                                                                                 |
|                                                                                                                                    | <p>When <code>{Blind}</code> is the first item in the string, the program avoids releasing Alt/Control/Shift/Win if they started out in the down position. For example, the hotkey <code>+s::Send {Blind}abc</code> would send ABC rather than abc because the user is holding down the Shift key.</p> <p><code>{Blind}</code> also causes <a href="#">SetStoreCapslockMode</a> to be ignored; that is, the state of Capslock is not changed. Finally, <code>{Blind}</code> omits the extra Control keystrokes that would otherwise be sent; such keystrokes</p>    |

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>{Blind}</b> | <p>prevent: 1) Start Menu appearance during LWin/RWin keystrokes; 2) menu bar activation during Alt keystrokes.</p> <p>Blind-mode is used internally when <a href="#">remapping a key</a>. For example, the remapping a:b would produce: 1) "b" when you type "a"; 2) uppercase B when you type uppercase A; and 3) Control-B when you type Control-A.</p> <p>{Blind} is not supported by <a href="#">SendRaw</a> and <a href="#">ControlSendRaw</a>. Furthermore, it is not completely supported by <a href="#">SendPlay</a>, especially when dealing with the modifier keys (Control, Alt, Shift, and Win).</p> |
| <b>{Raw}</b>   | <p>Enables <i>Raw mode</i>, which causes the following characters to be interpreted literally: <code>^+!#{}</code>. Although the string <code>{Raw}</code> need not occur at the beginning of the string, once specified, it stays in effect for the remainder of the string.</p>                                                                                                                                                                                                                                                                                                                                 |

## Repeating or Holding Down a Key

**To repeat a keystroke:** Enclose in braces the name of the key followed by the number of times to repeat it. For example:

```
Send {DEL 4} ; Presses the Delete key 4 times.
Send {S 30} ; Sends 30 uppercase S
characters.
Send +{TAB 4} ; Presses Shift-Tab 4 times.
```

**To hold down or release a key:** Enclose in braces the name of the key followed by the word **Down** or **Up**. For example:

```
Send {b down}{b up}
Send {TAB down}{TAB up}
Send {Up down} ; Press down the up-arrow key.
Sleep 1000 ; Keep it down for one second.
Send {Up up} ; Release the up-arrow key.
```

When a key is held down via the method above, it does not begin auto-repeating like it would if you were physically holding it down (this is because auto-repeat is a driver/hardware feature). However, a [Loop](#) can be used to simulate auto-repeat. The following example sends 20 tab keystrokes:

```
Loop 20
{
 Send {Tab down} ; Auto-repeat consists of
consecutive down-events (with no up-events).
 Sleep 30 ; The number of milliseconds
between keystrokes (or use SetKeyDelay).
```

```
}
Send {Tab up} ; Release the key.
```

The word *DownTemp* may also be used. Its effect is the same as *Down* except for the modifier keys (Control/Shift/Alt/Win). In those cases, *DownTemp* tells subsequent sends that the key is not permanently down, and may be released whenever a keystroke calls for it. For example, `Send {Control`  
`DownTemp}` followed later by `Send a` would produce a normal "a" keystroke, not a control-A keystroke.

## General Remarks

There are no specific limitations on which characters the Send command supports. If a character does not exist in the current keyboard layout, it is simulated by sending a Unicode character packet or `Alt+nnnn` combination, depending on the version of AutoHotkey. Unicode characters are supported directly as text in Unicode versions of AutoHotkey, or using `{U+nnnn}` notation in any version.

**BlockInput Compared to SendInput/SendPlay:** Although the `BlockInput` command can be used to prevent any keystrokes physically typed by the user from disrupting the flow of simulated keystrokes, it is often better to use `SendInput` or `SendPlay` so that keystrokes and mouse clicks become uninterruptible. This is because unlike `BlockInput`, `SendInput/Play` does not discard what the user types during the send; instead, such keystrokes are buffered and sent afterward.

When sending a large number of keystrokes, a `continuation section` can be used to improve readability and maintainability.

Since the operating system does not allow simulation of the CTRL-ALT-DELETE combination, doing something like `Send ^!{Delete}` will have no effect.

**Send may have no effect** on Windows Vista or later if the active window is running with administrative privileges and the script is not. This is due to a security mechanism called User Interface Privilege Isolation.

## SendInput

SendInput is generally the preferred method to send keystrokes and mouse clicks because of its superior speed and reliability. Under most conditions, SendInput is nearly instantaneous, even when sending long strings. Since SendInput is so fast, it is also more reliable because there is less opportunity for some other window to pop up unexpectedly and intercept the keystrokes. Reliability is further improved by the fact that anything the user types during a SendInput is postponed until afterward.

Unlike the other sending modes, the operating system limits SendInput to about 5000 characters (this may vary depending on the operating system's version and performance settings). Characters and events beyond this limit are not sent.

**Note:** SendInput ignores SetKeyDelay because the operating system does not support a delay in this mode. However, when SendInput reverts to [SendEvent](#) under the conditions described below, it uses `SetKeyDelay -1, 0` (unless SendEvent's KeyDelay is `-1, -1`, in which case `-1, -1` is used). When SendInput reverts to [SendPlay](#), it uses SendPlay's KeyDelay.

If a script *other than* the one executing SendInput has a [low-level keyboard hook](#) installed, SendInput automatically reverts to [SendEvent](#) (or [SendPlay](#) if `SendMode InputThenPlay` is in effect). This is done because the presence of an external hook disables all of SendInput's advantages, making it inferior to both SendPlay and SendEvent. However, since SendInput is unable to detect a low-level hook in programs other than AutoHotkey v1.0.43+, it will not revert in

these cases, making it less reliable than SendPlay/Event.

When SendInput sends mouse clicks by means such as `{Click}`, and `CoordMode Mouse, Relative` is in effect (the default), every click will be relative to the window that was active at the start of the send. Therefore, if SendInput intentionally activates another window (by means such as alt-tab), the coordinates of subsequent clicks within the same command will be wrong because they will still be relative to the old window rather than the new one.

## SendPlay

**Note:** SendPlay may have no effect at all if UAC is enabled, even if the script is running as an administrator. For more information, refer to the [FAQ](#).

SendPlay's biggest advantage is its ability to "play back" keystrokes and mouse clicks in a broader variety of games than the other modes. For example, a particular game may accept [hotstrings](#) only when they have the [SendPlay](#) option.

Of the three sending modes, SendPlay is the most unusual because it does not simulate keystrokes and mouse clicks per se. Instead, it creates a series of events (messages) that flow directly to the active window (similar to [ControlSend](#), but at a lower level). Consequently, SendPlay does not trigger hotkeys or hotstrings.

Like [SendInput](#), SendPlay's keystrokes do not get interspersed with keystrokes typed by the user. Thus, if the user happens to type something during a SendPlay, those keystrokes are postponed until afterward.

Although SendPlay is considerably slower than SendInput, it is usually faster than the traditional [SendEvent](#) mode (even when [KeyDelay](#) is -1).

The Windows keys (LWin and RWin) are automatically blocked during a SendPlay if the [keyboard hook](#) is installed. This prevents the Start Menu from appearing if the user accidentally presses a Windows key during the send. By contrast, keys other than LWin and RWin do not need to be blocked because the operating system automatically postpones them until after the SendPlay (via buffering).

SendPlay does not use the standard settings of SetKeyDelay and SetMouseDelay. Instead, it defaults to no delay at all, which can be changed as shown in the following examples:

```
SetKeyDelay, 0, 10, Play ; Note that both 0
and -1 are the same in SendPlay mode.
SetMouseDelay, 10, Play
```

SendPlay is unable to turn on or off the Capslock, Numlock, or Scroll-lock keys. Similarly, it is unable to change a key's state as seen by GetKeyState unless the keystrokes are sent to one of the script's own windows. Even then, any changes to the left/right modifier keys (e.g. RControl) can be detected only via their neutral counterparts (e.g. Control). Also, SendPlay has other limitations described on the [SendMode](#) page.

Unlike [SendInput](#) and [SendEvent](#), the user may interrupt a SendPlay by pressing Control-Alt-Del or Control-Escape. When this happens, the remaining keystrokes are not sent but the script continues executing as though the SendPlay had completed normally.

Although SendPlay can send LWin and RWin events, they are sent directly to the active window rather than performing their native operating system function. To work around this, use [SendEvent](#). For example, `SendEvent #r` would show the Start Menu's Run dialog.

## Related

[SendMode](#), [SetKeyDelay](#), [SetStoreCapslockMode](#), [Escape sequences \(e.g. ^%\)](#), [ControlSend](#), [BlockInput](#), [Hotstrings](#), [WinActivate](#)

## Examples

Send Sincerely,{enter}John Smith ; Types a two-line signature.

Send !fs ; Select the File->Save menu (Alt+F followed by S).

Send {End}+{Left 4} ; Jump to the end of the text then send four shift+left-arrow keystrokes.

SendInput {Raw}A long series of raw characters sent via the fastest method (SendInput).

# SendLevel

Controls which artificial keyboard and mouse events are ignored by hotkeys and hotstrings.

**SendLevel** Level

|                 |                 |              |
|-----------------|-----------------|--------------|
| <b>Command</b>  | <b>Example:</b> | SendLevel 1  |
| <b>Function</b> | <b>Example:</b> | SendLevel(1) |

## Parameters

### Level

An integer between 0 and 100.

## General Remarks

By default, [hook hotkeys](#) and [hotstrings](#) ignore keyboard and mouse events generated by any AutoHotkey script. In some cases it can be useful to override this behaviour; for instance, to allow a remapped key to be used to trigger other hotkeys. SendLevel and [#InputLevel](#) provide the means to achieve this.

SendLevel sets the level for events generated by the current [script thread](#), while [#InputLevel](#) sets the level for any hotkeys or hotstrings beneath it. For any event generated by a script to trigger a hook hotkey or hotstring, the send level of the event must be higher than the input level of the hotkey or hotstring.

## Compatibility:

- `SendPlay` is not affected by `SendLevel`.
- `SendInput` is affected by `SendLevel`, but the script's own hook hotkeys cannot be activated while a `SendInput` is in progress, since it temporarily deactivates the hook.
- Hotkeys using the "reg" method are incapable of distinguishing physical and artificial input, so are not affected by `SendLevel`. However, hotkeys above level 0 always use the keyboard or mouse hook.
- Auto-replace hotstrings always generate keystrokes at level 0, since it is usually undesirable for the replacement text to trigger another hotstring or hotkey. To work around this, use a non-auto-replace hotstring and the `Send` command.
- Any character which does not correspond to a key in the current keyboard layout cannot trigger a hotstring, even if `SendLevel` is used. This is a limitation of the hotstring recognizer.

The built-in variable `A_SendLevel` contains the current setting.

Every newly launched hotkey or hotstring thread starts off with a send level equal to the input level of the hotkey or hotstring. Every other newly launched thread (such as a custom menu item or timed subroutine) starts off fresh with the default setting, which is typically 0 but may be changed by using this command in the auto-execute section.

If `SendLevel` is used in the auto-execute section, it also affects keyboard and mouse remapping.

AutoHotkey versions older than v1.1.06 behave as though `#InputLevel 0` and `SendLevel 0` are in effect.

## Related

[#InputLevel](#), [Send](#), [Click](#), [MouseClick](#), [MouseClickDrag](#)

## Examples

```
SendLevel 1
Send btw{Space} ; Produces "by the way ".

; This may be defined in a separate script:
::btw::by the way
```

# SendMode

Makes `Send` synonymous with `SendEvent` or `SendPlay` rather than the default (`SendInput`). Also makes `Click` and `MouseMove/Click/Drag` use the specified method.

**SendMode** Input | Play | Event | InputThenPlay

```
Command Example: SendMode "Input"
Function Example: SendMode("Input")
```

The first parameter is one of the following words:

**Event:** Switches to the `SendEvent` method for `Send`, `SendRaw`, `Click`, and `MouseMove/Click/Drag`.

**Input:** This is the starting default used by all scripts. It uses the `SendInput` method for `Send`, `SendRaw`, `Click`, and `MouseMove/Click/Drag`. Known limitations:

- Windows Explorer ignores `SendInput`'s simulation of certain navigational hotkeys such as `Alt+LeftArrow`. To work around this, use either `SendEvent !{Left}` or `SendInput {Backspace}`.

**InputThenPlay:** Same as above except that rather than falling back to `Event` mode when `SendInput` is `unavailable`, it reverts to `Play` mode (below). This also causes the `SendInput` command itself to revert to `Play` mode when `SendInput` is

unavailable.

**Play:** Switches to the `SendPlay` method for `Send`, `SendRaw`, `Click`, and `MouseMove/Click/Drag`.

Known limitations:

- Characters that do not exist in the current keyboard layout (such as Ô in English) cannot be sent. To work around this, use `SendEvent`.
- Simulated mouse dragging might have no effect in RichEdit controls (and possibly others) such as those of WordPad and Metapad. To use an alternate mode for a particular drag, follow this example: `SendEvent {Click 6, 52, down}{Click 45, 52, up}`.
- Simulated mouse wheel rotation produces movement in only one direction (usually downward, but upward in some applications). Also, wheel rotation might have no effect in applications such as MS Word and Notepad. To use an alternate mode for a particular rotation, follow this example: `SendEvent {WheelDown 5}`.
- When using `SendMode Play` in the auto-execute section (top part of the script), all remapped keys are affected and might lose some of their functionality. See [SendPlay remapping limitations](#) for details.
- `SendPlay` does not trigger AutoHotkey's hotkeys or hotstrings, or global hotkeys registered by other programs or the OS.

## Remarks

Since `SendMode` also changes the mode of `Click` and `MouseMove/Click/Drag`, there may be times when you wish to use a different mode for a particular mouse

event. The easiest way to do this is via `{Click}`. For example:

```
SendEvent {Click 100, 200} ; SendEvent uses
the older, traditional method of clicking.
```

If `SendMode` is used in the auto-execute section (top part of the script), it also affects [keyboard and mouse remapping](#). In particular, if you use `SendMode Play` with remapping, see [SendPlay remapping limitations](#).

The built-in variable `A_SendMode` contains the current setting.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[Send](#), [SetKeyDelay](#), [SetMouseDelay](#), [Click](#), [MouseClicked](#), [MouseClickedDrag](#), [MouseMove](#)

## Examples

```
SendMode Input
SendMode InputThenPlay
```

# SetDefaultMouseSpeed

Sets the mouse speed that will be used if unspecified in [Click](#) and [MouseMove/Click/Drag](#).

**SetDefaultMouseSpeed** Speed

|                 |                 |                           |
|-----------------|-----------------|---------------------------|
| <b>Command</b>  | <b>Example:</b> | SetDefaultMouseSpeed 100  |
| <b>Function</b> | <b>Example:</b> | SetDefaultMouseSpeed(100) |

## Parameters

### Speed

The speed to move the mouse in the range 0 (fastest) to 100 (slowest).  
Note: a speed of 0 will move the mouse instantly.

## Remarks

SetDefaultMouseSpeed is ignored for [SendInput/Play modes](#); they move the mouse instantaneously (however, [SetMouseDelay](#) has a mode that applies to SendPlay). To visually move the mouse more slowly -- such as a script that performs a demonstration for an audience -- use `SendEvent {Click 100, 200}` or `SendMode Event` (optionally in conjunction with [BlockInput](#)).

If this command is not used, the default mouse speed is 2. The built-in variable `A_DefaultMouseSpeed` contains the current setting.

The commands [MouseClicked](#), [MouseMove](#), and [MouseClickedDrag](#) all have a parameter to override the default mouse speed.

Whenever *Speed* is greater than zero, [SetMouseDelay](#) also influences the speed by producing a delay after each incremental move the mouse makes toward its destination.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[SetMouseDelay](#), [SendMode](#), [Click](#), [MouseClicked](#), [MouseMove](#), [MouseClickedDrag](#), [SetWinDelay](#), [SetControlDelay](#), [SetKeyDelay](#), [SetKeyDelay](#)

## Example

```
SetDefaultMouseSpeed, 0 ; Move the mouse
instantly.
```

# SetKeyDelay

Sets the delay that will occur after each keystroke sent by [Send](#) and [ControlSend](#).

```
SetKeyDelay [Delay, PressDuration, Play]
```

```
Command Example: SetKeyDelay 100
Function Example: SetKeyDelay(100)
```

## Parameters

### Delay

Time in milliseconds. Use -1 for no delay at all and 0 for the smallest possible delay (however, if the *Play* parameter is present, both 0 and -1 produce no delay). Leave this parameter blank to retain the current *Delay*.

If `SetKeyDelay` is never used by a script, the default *Delay* for the traditional `SendEvent` mode is 10. For [SendPlay mode](#), the default *Delay* is -1. The default *PressDuration* (below) is -1 for both modes.

### PressDuration

Certain games and other specialized applications may require a delay inside each keystroke; that is, after the press of the key but before its release.

Use -1 for no delay at all (default) and 0 for the smallest possible delay (however, if the *Play* parameter is present, both 0 and -1 produce no

delay). Omit this parameter to leave the current *PressDuration* unchanged.

Note: *PressDuration* also produces a delay after any change to the modifier key state (CTRL, ALT, SHIFT, and WIN) needed to support the keys being sent.

### Play

The word *Play* applies the above settings to the *SendPlay mode* rather than the traditional *SendEvent mode*. If a script never uses this parameter, the delay is always -1/-1 for *SendPlay*.

## Remarks

**Note:** *SetKeyDelay* is not obeyed by *SendInput*; there is no delay between keystrokes in that mode. This same is true for *Send* when *SendMode Input* is in effect.

A short delay (sleep) is done automatically after every keystroke sent by *Send* or *ControlSend*. This is done to improve the reliability of scripts because a window sometimes can't keep up with a rapid flood of keystrokes.

Due to the granularity of the OS's time-keeping system, delays might be rounded up to the nearest multiple of 10 or 15. For example, a delay between 1 and 10 (inclusive) is equivalent to 10 or 15 on most Windows XP systems (and probably 2k).

For *Send/SendEvent mode*, a delay of 0 internally executes a *Sleep(0)*, which

yields the remainder of the script's timeslice to any other process that may need it. If there is none, `Sleep(0)` will not sleep at all. By contrast, a delay of `-1` will never sleep. For better reliability, `0` is recommended as an alternative to `-1`.

When the delay is set to `-1`, a script's process-priority becomes an important factor in how fast it can send keystrokes when using the traditional `SendEvent mode`. To raise a script's priority, use `ProcessSetPriority, High`. Although this typically causes keystrokes to be sent faster than the active window can process them, the system automatically buffers them. Buffered keystrokes continue to arrive in the target window after the `Send` command completes (even if the window is no longer active). This is usually harmless because any subsequent keystrokes sent to the same window get queued up behind the ones already in the buffer.

The built-in variable `A_KeyDelay` contains the current setting of *Delay* for `Send/SendEvent` mode. `A_KeyDuration` contains the setting for *PressDuration*, while `A_KeyDelayPlay` and `A_KeyDurationPlay` contain the settings for `SendPlay`.

Every newly launched thread (such as a hotkey, custom menu item, or timed subroutine) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

`Send`, `ControlSend`, `SendMode`, `SetMouseDelay`, `SetControlDelay`,

SetWinDelay, Click

## Example

```
SetKeyDelay, 0
```

# SetMouseDelay

Sets the delay that will occur after each mouse movement or click.

```
SetMouseDelay Delay [, Play]
```

```
Command Example: SetMouseDelay 100
Function Example: SetMouseDelay(100)
```

## Parameters

### Delay

Time in milliseconds. Use -1 for no delay at all and 0 for the smallest possible delay (however, if the *Play* parameter is present, both 0 and -1 produce no delay). If unset, the default delay is 10 for the traditional SendEvent mode and -1 for [SendPlay mode](#).

### Play

The word *Play* applies the delay to the [SendPlay mode](#) rather than the traditional Send/SendEvent mode. If a script never uses this parameter, the delay is always -1 for SendPlay.

## Remarks

A short delay (sleep) is done automatically after every mouse movement or click generated by [Click](#) and [MouseMove/Click/Drag](#) (except for [SendInput mode](#)).

This is done to improve the reliability of scripts because a window sometimes can't keep up with a rapid flood of mouse events.

Due to the granularity of the OS's time-keeping system, delays might be rounded up to the nearest multiple of 10 or 15. For example, a delay between 1 and 10 (inclusive) is equivalent to 10 or 15 on most Windows XP systems (and probably 2k).

A delay of 0 internally executes a Sleep(0), which yields the remainder of the script's timeslice to any other process that may need it. If there is none, Sleep(0) will not sleep at all. By contrast, a delay of -1 will never sleep.

The built-in variable **A\_MouseDelay** contains the current setting for Send/SendEvent mode. **A\_MouseDelayPlay** contains the current setting for SendPlay mode.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[SetDefaultMouseSpeed](#), [Click](#), [MouseMove](#), [MouseClicked](#), [MouseClickedDrag](#), [SendMode](#), [SetKeyDelay](#), [SetControlDelay](#), [SetWinDelay](#)

## Example



# SetCapsLockState/SetNumLockState/Set

Sets the state of the Capslock/NumLock/ScrollLock key. Can also force the key to stay on or off.

**SetCapsLockState** [State]

**SetNumLockState** [State]

**SetScrollLockState** [State]

```
Command Example: SetScrollLockState "Off"
Function Example: SetScrollLockState("Off")
```

## Parameters

### State

If this parameter is omitted, the AlwaysOn/Off attribute of the key is removed (if present). Otherwise, specify one of the following words:

**On** or 1 (*true*): Turns on the key and removes the AlwaysOn/Off attribute of the key (if present).

**Off** or 0 (*false*): Turns off the key and removes the AlwaysOn/Off attribute of the key (if present).

**AlwaysOn**: Forces the key to stay on permanently.

**AlwaysOff:** Forces the key to stay off permanently.

## Remarks

A key can also be toggled to its opposite state via the [Send](#) command; for example: `Send {Capslock}`.

Keeping a key *AlwaysOn* or *AlwaysOff* requires the [keyboard hook](#), which will be automatically installed in such cases.

## Related

[SetStoreCapslockMode](#), [GetKeyState](#)

## Examples

```
SetNumlockState, on
SetScrollLockState, AlwaysOff
```

# SetStoreCapslockMode

Whether to restore the state of CapsLock after a [Send](#).

**SetStoreCapslockMode** "On|Off"

## Parameters

### On|Off

**On** or 1 ([true](#)): This is the initial setting for all scripts: The CapsLock key will be restored to its former value if [Send](#) needed to change it temporarily for its operation.

**Off** or 0 ([false](#)): The state of the CapsLock key is not changed at all. As a result, [Send](#) will invert the case of the characters if Capslock happens to be ON during the operation.

## Remarks

This means that the Capslock key will not always be turned off for [Send](#) and [ControlSend](#). Even when it is successfully turned off, it might not get turned back on after the keys are sent.

This command is rarely used because the default behavior is best in most cases.

The built-in variable **A\_StoreCapslockMode** contains the current setting.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[SetCaps/Num/ScrollLockState](#), [Send](#), [ControlSend](#)

## Example

```
SetStoreCapslockMode, off
```

# Cast

Converts a value from one data type to another data type.

```
OutputVar := Cast(DataType, VarOrValue, NewDataType)
```

```
Function Example: NewValue :=
Cast("float", 2.5, "UInt")
```

## Parameters

### OutputVar

The name of the variable in which to store the converted value.

### DataType

Data type of given variable or value, must be one of the following strings:  
UInt, Int, Int64, Short, UShort, Char, UChar, Double, Float, Ptr or UPtr

### VarOrValue

The variable or a value to be converted.

### NewDataType

Data type to convert to, must be one of the strings as in DataType.

## Examples

```
MsgBox Cast("float", 1.4, "int")
```



# Download

Downloads a file from the Internet.

```
Download(URL, FileName)
```

```
Download("http://ahkscript.org/download/index.htm", A_ScriptDir "\ahkscript.htm")
```

```
Download,
http://ahkscript.org/download/index.htm,
%A_ScriptDir%\ahkscript.htm
```

## Parameters

### URL

URL of the file to download. For example, `http://someorg.org` might retrieve the welcome page for that organization.

### Filename

**Download to a file:** Specify the name of the file to be created locally, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. Any existing file will be **overwritten** by the new file.

**Download to a variable:** See the example below. For alternative

methods, see the following forum topic:

[www.autohotkey.com/forum/topic10466.html](http://www.autohotkey.com/forum/topic10466.html)

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

The download might appear to succeed even when the remote file doesn't exist. This is because many web servers send an error page instead of the missing file. This error page is what will be saved in place of *Filename*.

Firewalls or the presence of multiple network adapters may cause this function to fail. Also, some websites may block such downloads.

**Caching:** By default, the URL is retrieved directly from the remote server (that is, never from Internet Explorer's cache). To permit caching, precede the URL with \*0 followed by a space; for example: \*0 <http://someorg.org>. The zero following the asterisk may be replaced by any valid `dwFlags` number; for details, search [www.microsoft.com](http://www.microsoft.com) for `InternetOpenUrl`.

**Proxies:** If a proxy server has been configured in Microsoft Internet Explorer's settings, it will be used.

**FTP and Gopher** URLs are also supported. For example:

```
Download, ftp://example.com/home/My File.zip,
C:\My Folder\My File.zip ; Log in anonymously.
```

```
Download,
ftp://user:pass@example.com:21/home/My
File.zip, C:\My Folder\My File.zip ; Log in as
a specific user.
Download, ftp://user:pass@example.com/My
Directory, C:\Dir Listing.html ; Gets a
directory listing in HTML format.
```

## Related

[FileRead](#), [FileCopy](#)

## Examples

```
Download,
http://autohotkey.com/download/2.0/version.txt,
C:\AutoHotkey Latest Version.txt
Download, http://someorg.org/archive.zip,
C:\SomeOrg's Archive.zip
```

```
; Example: Download text to a variable:
whr := ComObjCreate("WinHttp.WinHttpRequest.5.1")
whr.Open("GET",
"http://autohotkey.com/download/2.0/version.txt")
whr.Send()
; Using 'true' above and the call below allows the
script to remain responsive.
whr.WaitForResponse()
version := whr.ResponseText
MsgBox % version
```

```
; Example: Make an asynchronous HTTP request.
```

```

req := ComObjCreate("Msxml2.XMLHTTP")
; Open a request with async enabled.
req.open("GET",
"https://autohotkey.com/download/2.0/version.txt",
true)
; Set our callback function (v1.1.17+).
req.onreadystatechange := Func("Ready")
; Send the request. Ready() will be called when
it's complete.
req.send()
/*
; If you're going to wait, there's no need for
onreadystatechange.
; Setting async=true and waiting like this allows
the script to remain
; responsive while the download is taking place,
whereas async=false
; will make the script unresponsive.
while req.readyState != 4
 sleep 100
*/
#Persistent

Ready() {
 global req
 if (req.readyState != 4) ; Not done yet.
 return
 if (req.status == 200 || req.status == 304) ;
OK.
 MsgBox % "Latest AutoHotkey version: "
req.responseText
 else
 MsgBox("Status " req.status,, 16)
 ExitApp
}

```

# Edit

Opens the current script for editing in the associated editor.

## Edit

The Edit command opens the current script for editing using the associated "edit" verb in the registry (or Notepad if no verb). However, if an editor window appears to have the script already open (based on its window title), that window is activated instead of opening a new instance of the editor.

This command has no effect when operating from within a compiled script.

On a related note, AutoHotkey syntax highlighting can be enabled for various editors - see below. In addition, context sensitive help for AutoHotkey commands can be enabled in any editor via [this example](#). Finally, your productivity may be improved by using an auto-completion utility like [ISense](#), which works in almost any editor. It watches what you type and displays menus and parameter lists, which does some of the typing for you and reminds you of the order of parameters.

## Related

[Reload](#)

## Example

Edit ; opens the script for editing.

```
; If your editor's command-line usage is something
like the following,
; this script can be used to set it as the default
editor for ahk files:
```

```
;
; Editor.exe "Full path of script.ahk"
```

```
;
; When you run the script, it will prompt you to
select the executable
; file of your editor.
```

```
;
FileSelect Editor, 2,, Select your editor,
Programs (*.exe)
if ErrorLevel
 ExitApp
RegWrite REG_SZ,
HKCR\AutoHotkeyScript\Shell\Edit\Command,,
"%Editor%" "%1"
```

## Editors with AutoHotkey Support

**SciTE4AutoHotkey** is a custom version of the text editor known as SciTE, tailored for editing AutoHotkey scripts. Its features include:

- Syntax highlighting
- Smart auto-indent
- Auto-complete
- Calltips (also known as IntelliSense)
- Code folding

- Support for [interactive debugging](#)
- Other tools for AutoHotkey scripting

SciTE4AutoHotkey can be found here: <http://fincs.ahk4.net/scite4ahk/>

**AHK Studio** is a script editor built using AutoHotkey, for editing AutoHotkey scripts. See the following forum thread for details, demonstration videos and an ever-growing list of features: <https://autohotkey.com/boards/viewtopic.php?t=300>

**AutoGUI** is an integrated development environment for AutoHotkey which combines a GUI designer with a script editor. It can be found here: <https://autohotkey.com/boards/viewtopic.php?f=6&t=10157>

**Other editors** for which AutoHotkey syntax highlighting can be enabled:

- AkelPad - <http://www.autohotkey.com/forum/topic23586.html>
- Crimson Editor - <http://www.autohotkey.com/forum/topic5506.html>
- Eclipse, FAR manager, and any other editors which use Colorer take5 - <http://www.autohotkey.com/forum/topic10378.html>
- Emacs - <https://github.com/tinku99/ahk-org-mode>
- Notepad++ - <http://www.autohotkey.com/forum/topic58792.html>
- Notepad2 - <http://www.autohotkey.com/forum/viewtopic.php?t=37652>
- PSPad - <http://www.autohotkey.com/forum/topic9294.html>
- SciTE and possibly other Scintilla based editors - <http://www.autohotkey.com/forum/topic9656.html>
- Sublime Text Editor - <http://www.autohotkey.com/forum/viewtopic.php?>

[p=368326#368326](#)

- Total Commander with Synplus plugin - <http://www.autohotkey.com/forum/topic7278.html>

Additionally, the zip download of AutoHotkey Basic (<http://www.autohotkey.com/download/1.0/>) includes files for enabling syntax highlighting in the following editors. However, some of these files are badly out of date and may or may not work:

- ConTEXT
- EditPlus
- EmEditor
- jEdit
- MED
- TextPad
- UltraEdit
- Vim

If your favourite editor isn't listed here, try your luck by searching the [forums](#).

To get an editor added to this page, contact Lexikos [via the forums](#) or [GitHub](#).

# Func()

Retrieves a reference to the specified function.

```
FunctionReference := Func(FunctionName)
```

## Parameters

### FunctionName

The name of the function whose reference is retrieved. *FunctionName* must exist explicitly in the script.

## Return Value

This function returns a [reference to \*FunctionName\*](#). If *FunctionName* does not exist explicitly in the script (by means such as [#Include](#) or a non-dynamic call to a [library function](#)), it returns 0.

## Remarks

This function can be used to call the function or retrieve [information](#) such as the minimum and maximum number of parameters.

## Related

[Function References](#), [Func Object](#)

## Examples

```
; Retrieve a reference to the function named
"StrLen".
```

```
fn := Func("StrLen")
```

```
; Display information about the function.
```

```
MsgBox % fn.Name "()" is " (fn.IsBUILTIn ? "built-
in." : "user-defined.")
```

# IsByRef()

Returns a non-zero number if a ByRef parameter of a function was supplied with the specified variable.

```
TrueOrFalse := IsByRef(UnquotedVarName)
```

## Parameters

### UnquotedVarName

The name of the variable (*not in quotes*). For example:

```
IsByRef(MyVar).
```

## Return Value

This function returns 1 if *UnquotedVarName* is a [ByRef parameter](#) and the caller supplied a variable; or 0 if *UnquotedVarName* is any other kind of variable.

## Related

[ByRef parameters](#)

## Examples

```
MsgBox, % Function(MyVar)

Function(ByRef Param)
{
```



# IsFunc()

Returns a non-zero number if the specified function exists in the script.

```
MinParamsPlus1 := IsFunc(FunctionName)
```

## Parameters

### FunctionName

The name of the function whose minimum number of parameters is retrieved. *FunctionName* must exist explicitly in the script.

*FunctionName* can be a [function reference](#) instead of a name.

## Return Value

This function returns one plus the minimum number of parameters (e.g. 1 for a function that requires zero parameters, 2 for a function that requires 1 parameter, etc.). If *FunctionName* does not exist explicitly in the script (by means such as [#Include](#) or a non-dynamic call to a [library function](#)), it returns 0.

## Related

[Dynamically Calling a Function](#), [Function References](#), [Func Object](#), [Func](#), [A\\_ThisFunc](#)

## Examples

```
count := IsFunc("RegExReplace") ; Any function
name can used here.
if count
 MsgBox, % "This function exists and has " count-
1 " mandatory parameters."
else
 MsgBox, % "This function does not exist."
```

# IsLabel()

Returns a non-zero number if the specified label exists in the script.

```
TrueOrFalse := IsLabel(LabelName)
```

## Parameters

### LabelName

The name of a [subroutine](#), [hotkey](#), or [hotstring](#) (do not include the trailing colon(s) in *LabelName*).

## Return Value

This function returns a non-zero number if *LabelName* exists in the script.

## Remarks

This function is useful to avoid runtime errors when specifying a dynamic label in commands such as [Gosub](#), [Hotkey](#), [Menu](#), and [Gui.Add](#).

## Related

[Labels](#)

## Examples



# IsObject()

Returns a non-zero number if the specified value is an object.

```
TrueOrFalse := IsObject(ObjectValue)
```

## Parameters

### ObjectValue

A [object](#) stored in a variable, returned from a function, stored in another object or written directly.

## Return Value

This function returns 1 if *ObjectValue* is an object; otherwise 0.

## Related

[Objects](#)

## Examples

```
object := {key: "value"}

if IsObject(object)
 MsgBox, This is an object.
else
 MsgBox, This is not an object.
```

# ListLines

Enables or disables line logging or displays the script lines most recently executed.

```
ListLines ["On|Off"]
```

## Parameters

### On|Off

If blank or omitted, the history of lines most recently executed is shown. The first parameter affects only the behavior of the [current thread](#) as follows:

**On** or 1 ([true](#)): Includes subsequently-executed lines in the history.

**Off** or 0 ([false](#)): Omits subsequently-executed lines from the history. This is the starting default for all scripts.

## Remarks

ListLines (with no parameter) is equivalent to selecting the "View->Lines most recently executed" menu item in the main window. It can help [debug a script](#). However, since line logging is disabled by default, "ListLines On" must be used before executing the section of code which is to be debugged.

ListLines Off/On can be used to selectively omit some lines from the

history, which can help prevent the history from filling up too quickly (such as in a loop with many fast iterations). Additionally, performance is reduced by a few percent while line logging is enabled.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

Although there is no built-in variable "A\_ListLines", similar functionality can be achieved by including the following in a script:

```
ListLines(PassTrueToTurnOnOrFalseToTurnOff) ;
Returns the previous setting of ListLines
(prior to this call).
{
 static sListLines := true ; The starting
default for all scripts is "ListLines On".
 ListLines %
PassTrueToTurnOnOrFalseToTurnOff ? "On" : "Off"
; Execute ListLines unconditionally to omit the
lines executed below from the log.
 ListLines_prev := sListLines
 sListLines :=
PassTrueToTurnOnOrFalseToTurnOff
 return ListLines_prev
}

; To use the above function:
prev_ListLines := ListLines(false) ; Turn off
ListLines temporarily.
; ...
ListLines(prev_ListLines) ; Restore ListLines
```

to its previous setting.

On a related note, the built-in variables [A\\_LineNumber](#) and [A\\_LineFile](#) contain the currently executing line number and the file name to which it belongs.

## Related

[KeyHistory](#), [ListHotkeys](#), [ListVars](#)

## Example

```
ListLines
ListLines Off
```

# ListVars

Displays the script's [variables](#): their names and current contents.

## ListVars

```
Command Example: ListVars
Function Example: ListVars()
```

## Remarks

This command is equivalent to selecting the "View->Variables" menu item in the main window. It can help you [debug a script](#).

Each line in the list consists of:

- 1) The name of the variable.
- 2) The length of the variable's contents and the variable's [capacity](#). For example:  
[50 of 63]
- 3) The first 60 characters of the variable's contents.

If this command is used inside a [function](#), the function's [local variables](#) will be listed first (above the script's global variables).

Known limitation: If a [function](#) (or the list of global variables itself) contains more than 10,000 variables, this command might not show them in exact alphabetical order; that is, some might be missing from the display.

## Related

[KeyHistory](#), [ListHotkeys](#), [ListLines](#)

## Example

### Function Syntax

```
var1 := "foo"
var2 := "bar"
obj := []
ListVars()
Pause()
```

### Command Syntax

```
var1 := "foo"
var2 := "bar"
obj := []
ListVars
Pause
```

# MonitorGet

Retrieves screen resolution and multi-monitor info.

```
Exists := MonitorGet([N, Left, Top, Right, Bottom])
Exists := MonitorGetWorkArea([N, Left, Top, Right, Bottom])
Count := MonitorGetCount()
Primary := MonitorGetPrimary()
Name := MonitorGetName([N])
```

## Parameters

N

The monitor number, between 1 and the number returned by `MonitorGetCount()`. If omitted, the primary monitor is used.

Left

Top

Right

Bottom

Output variables which receive bounding coordinates as described below.

Exists

Count

Primary

Name

An output variable or the return value of the command, as described below.

## MonitorGet

Checks for the existence of monitor number N and optionally retrieves its bounding coordinates. The information is stored in up to four variables passed as parameters. If N is too high or there is a problem retrieving the info, the variables are all made blank and the return value is false. For example:

```
if MonitorGet(2, Left, Top, Right, Bottom)
 MsgBox, Left: %Left% -- Top: %Top% --
 Right: %Right% -- Bottom %Bottom%.
else
 MsgBox, Monitor 2 doesn't exist or an error
 occurred.
```

## MonitorGetWorkArea

Same as the above except the area is reduced to exclude the area occupied by the taskbar and other registered desktop toolbars.

## MonitorGetCount

Retrieves the total number of monitors. Unlike the SM\_CMONITORS sub-command of [SysGet](#), the return value includes all monitors, even those not being used as part of the desktop.

## MonitorGetPrimary

Retrieves the number of the primary monitor, which will be 1 in a single-monitor system.

## MonitorGetName

Retrieves the operating system's name for monitor number N.

## Remarks

The built-in variables `A_ScreenWidth` and `A_ScreenHeight` contain the dimensions of the primary monitor, in pixels.

`SysGet` can be used to retrieve the bounding rectangle of all display monitors. For example, this retrieves the width and height:

```
SysGet, VirtualScreenWidth, 78
SysGet, VirtualScreenHeight, 79
MsgBox, %VirtualScreenWidth% x
%VirtualScreenHeight%
```

## Related

[DllCall](#), [WinGet](#), [SysGet](#)

## Examples

```
; This is a working script that displays info
about each monitor:
```

```
MonitorGetCount, MonitorCount
MonitorGetPrimary, MonitorPrimary
```



# OutputDebug

Sends a string to the debugger (if any) for display.

OutputDebug, Text

```
Command Example: OutPutDebug "Exit"
Function Example: OutPutDebug("Exit")
```

## Parameters

### Text

The text to send to the debugger for display. This text may include linefeed characters (`\n`) to start new lines. In addition, a single long line can be broken up into several shorter ones by means of a [continuation section](#).

## Remarks

If the script's process has no debugger, the system debugger displays the string. If the system debugger is not active, this command has no effect.

One example of a debugger is DebugView, which is free and available at [www.sysinternals.com](http://www.sysinternals.com).

See also: [other debugging methods](#)

## Related

[FileAppend](#), [continuation sections](#)

## Example

```
OutputDebug, %A_Now%: Because the window
"%TargetWindowTitle%" did not exist, the process
was aborted.
```

# Type

Returns the exact type of a value.

```
OutputVar := Type(Value)
```

## Return Value

The return value is a string indicating the type of *Value*. Possible values include:

- String
- Integer
- Float
- Object
- RegExMatchObject
- Func
- FileObject
- ComObject
- Struct

If *Value* is an object, the return value may be "Object" or a more specific built-in type such as "Func", "FileObject" or "ComObject".

## Remarks

This function typically shouldn't be used to determine if a value is numeric, since numeric *strings* are valid in math expressions and with most built-in functions.

However, in some cases the exact type of a value is more important. For instance, if `NumPut` is passed a variable containing a pure integer (not a numeric string), the integer will be used instead of the address of the variable.

To check if a value can be used as a number, use the expression `value is type`, where `type` is `"number"`, `"integer"` or `"float"`.

To check for any type of object, use the expression `value is 'object'`.

To check if an object is derived from a particular user-defined class, use `value is ClassName`.

## Related

[Variable types](#), [Expressions](#), [Value is Type](#)

## Examples

```
a := 1, b := 2.0, c := "3"
MsgBox % Type(a) ; Integer
MsgBox % Type(b) ; Float
MsgBox % Type(c) ; String
```

# Process

Functions for performing the following operations on a process: check if it exists; change its priority; close it; wait for it to exist; wait for it to close.

```
OutputVar := ProcessExist([PID-or-Name])
```

**Exist:** Sets *OutputVar* to the Process ID (PID) if a matching process exists, or 0 otherwise. If the *PID-or-Name* parameter is blank, the script's own PID is retrieved.

```
OutputVar := ProcessClose(PID-or-Name)
```

**Close:** If a matching process is successfully terminated, *OutputVar* is set to its former Process ID (PID). Otherwise (there was no matching process or there was a problem terminating it), it is set to 0. Since the process will be abruptly terminated -- possibly interrupting its work at a critical point or resulting in the loss of unsaved data in its windows (if it has any) -- this method should be used only if a process cannot be closed by using [WinClose](#) on one of its windows.

```
ProcessSetPriority Priority [, PID-or-Name]
```

**Priority:** Changes the priority (as seen in Windows Task Manager) of the first matching process to *Priority*. If the *PID-or-Name* parameter is blank, the script's own priority will be changed.

*Priority* should be one of the following letters or words: L (or Low), B (or BelowNormal), N (or Normal), A (or AboveNormal), H (or High), R (or Realtime). Note: Any process not designed to run at Realtime priority might reduce system stability if set to that level.

If called as a function, the return value is the Process ID (PID) of the process whose priority was set, or 0 if there is no matching process or there was a problem changing its priority.

```
OutputVar := ProcessWait(PID-or-Name [, Timeout])
```

**Wait:** Waits up to *Timeout* seconds (can contain a decimal point) for a matching process to exist. If *Timeout* is omitted, the command will wait indefinitely. If a matching process is discovered, *OutputVar* is set to its Process ID (PID). If the command times out, *OutputVar* is set to 0.

```
OutputVar := ProcessWaitClose(PID-or-Name [, Timeout])
```

**WaitClose:** Waits up to *Timeout* seconds (can contain a decimal point) for ALL matching processes to close. If *Timeout* is omitted, the command will wait indefinitely. If all matching processes are closed, *OutputVar* is set to 0. If the command times out, *OutputVar* is set to the Process ID (PID) of the first matching process that still exists.

## Parameters

*OutputVar*

The name of the variable in which to store the result, which is a Process ID (PID) or zero (0). When calling the command as a function, *OutputVar* represents its return value.

### PID-or-Name

This parameter can be either a number (the PID) or a process name as described below. It can also be left blank where indicated above.

**PID:** The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the [Run](#) command. Similarly, the PID of a window can be determined with [WinGet](#). [ProcessExist](#) can also be used to discover a PID.

**Name:** The name of a process is usually the same as its executable (without path), e.g. `notepad.exe` or `winword.exe`. Since a name might match multiple running processes, only the first process will be operated upon. The name is not case sensitive.

### Remarks

For *Wait* and *WaitClose*: Processes are checked every 100 milliseconds; the moment the condition is satisfied, the command stops waiting. In other words, rather than waiting for the timeout to expire, it immediately returns and continues execution of the script. Also, while the command is in a waiting state, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

**Process list:** Although there is no *ProcessList* function, the [examples section](#) demonstrates how to retrieve a list of processes via DllCall.

## Related

[Run](#), [WinGet](#), [WinClose](#), [WinKill](#), [WinWait](#), [WinWaitClose](#), [WinExist](#)

## Examples

```
; Example #1:
```

```
Run Notepad.exe, , , NewPID
ProcessSetPriority, High, %NewPID%
MsgBox The newly launched notepad's PID is
%NewPID%.
```

```
; Example #2:
```

```
ProcessWait, NewPID, Notepad.exe, 5.5
if NewPID = 0
{
 MsgBox The specified process did not appear
within 5.5 seconds.
 return
}
```

```
; Otherwise:
```

```
MsgBox A matching process has appeared (Process ID
is %NewPID%).
ProcessSetPriority, Low, %NewPID%
ProcessSetPriority, High ; Have the script set
itself to high priority.
```

```
WinClose Untitled - Notepad
ProcessWaitClose, ErrorLevel, %NewPID%, 5
if ErrorLevel ; The PID still exists.
 MsgBox The process did not close within 5
seconds.
```

**; Example #3: A hotkey to change the priority of the active window's process:**

```
#z:: ; Win+Z hotkey
Gui := GuiCreate(, "Set Priority")
Gui.Add("Text",, "
(
 Press ESCAPE to cancel, or double-click a new
 priority level for the following window:
)")
Gui.Add("Text", "wp", WinGetTitle("A"))
LB := Gui.Add("ListBox", "r5",
"Normal|High|Low|BelowNormal|AboveNormal")
SetPriority := Func("SetPriority").bind(LB,
WinGetPID("A"))
LB.OnEvent("DoubleClick", SetPriority)
Gui.Add("Button", "default",
"OK").OnEvent("Click", SetPriority)
Gui.OnEvent("Escape", "Gui_Close")
Gui.OnEvent("Close", "Gui_Close")
Gui.Show()
return

SetPriority(LB, PID)
{
 if ProcessSetPriority(LB.Text, PID)
 MsgBox("Success: Its priority was changed to
'%LB.Text%'.")
```

```

else
 MsgBox("Error: Its priority could not be
changed to '%LB.Text%'")
 WinClose()
}

Gui_Close(Gui)
{
 Gui.Destroy()
}

```

**; Example #4: Retrieves a list of running processes via DllCall then shows them in a MsgBox.**

```

d := " | " ; string separator
s := 4096 ; size of buffers and arrays (4 KB)

ScriptPID := ProcessExist() ; The PID of this
running script.
; Get the handle of this script with
PROCESS_QUERY_INFORMATION (0x0400)
h := DllCall("OpenProcess", "UInt", 0x0400, "Int",
false, "UInt", ScriptPID, "Ptr")
; Open an adjustable access token with this
process (TOKEN_ADJUST_PRIVILEGES = 32)
DllCall("Advapi32.dll\OpenProcessToken", "Ptr", h,
"UInt", 32, "PtrP", t)
VarSetCapacity(ti, 16, 0) ; structure of
privileges
NumPut(1, ti, 0, "UInt") ; one entry in the
privileges array...
; Retrieves the locally unique identifier of the
debug privilege:
DllCall("Advapi32.dll\LookupPrivilegeValue",

```

```

"Ptr", 0, "Str", "SeDebugPrivilege", "Int64P",
luid)
NumPut(luid, ti, 4, "Int64")
NumPut(2, ti, 12, "UInt") ; enable this
privilege: SE_PRIVILEGE_ENABLED = 2
; Update the privileges of this process with the
new access token:
r := DllCall("Advapi32.dll\AdjustTokenPrivileges",
"Ptr", t, "Int", false, "Ptr", &ti, "UInt", 0,
"Ptr", 0, "Ptr", 0)
DllCall("CloseHandle", "Ptr", t) ; close this
access token handle to save memory
DllCall("CloseHandle", "Ptr", h) ; close this
process handle to save memory

hModule := DllCall("LoadLibrary", "Str",
"Psapi.dll") ; increase performance by preloading
the library
s := VarSetCapacity(a, s) ; an array that
receives the list of process identifiers:
c := 0 ; counter for process identifiers
DllCall("Psapi.dll\EnumProcesses", "Ptr", &a,
"UInt", s, "UIntP", r)
Loop, % r // 4 ; parse array for identifiers as
DWORDs (32 bits):
{
 id := NumGet(a, A_Index * 4, "UInt")
 ; Open process with: PROCESS_VM_READ (0x0010) |
PROCESS_QUERY_INFORMATION (0x0400)
 h := DllCall("OpenProcess", "UInt", 0x0010 |
0x0400, "Int", false, "UInt", id, "Ptr")
 if !h
 continue
 VarSetCapacity(n, s, 0) ; a buffer that
receives the base name of the module:
 e := DllCall("Psapi.dll\GetModuleBaseName",
"Ptr", h, "Ptr", 0, "Str", n, "UInt", s//2)

```

```

 if !e ; fall-back method for 64-bit
processes when in 32-bit mode:
 if e :=
DllCall("Psapi.dll\GetProcessImageFileName",
"Ptr", h, "Str", n, "UInt", s//2)
 SplitPath %n%, n
 DllCall("CloseHandle", "Ptr", h) ; close
process handle to save memory
 if (n && e) ; if image is not null add to
list:
 l .= n . d, c++
}
DllCall("FreeLibrary", "Ptr", hModule) ; unload
the library to free memory
;Sort, l, C ; uncomment this line to sort the
list alphabetically
MsgBox("%l%", "%c% Processes", 0)

```

**; Example #5: Retrieves a list of running processes via COM.**

```

Gui := GuiCreate(, "Process List")
LV := Gui.Add("ListView", "x2 y0 w400 h500",
"Process Name|Command Line")
for process in
ComObjGet("winmgmts:").ExecQuery("Select * from
Win32_Process")
 LV.Add("", process.Name, process.CommandLine)
Gui.Show()

```

**; Win32\_Process: <http://msdn.microsoft.com/en-us/library/aa394372.aspx>**

# Run / RunWait

Runs an external program. Unlike Run, RunWait will wait until the program finishes before continuing.

```
Run Target [, WorkingDir, Max|Min|Hide|UseErrorLevel, OutputVarPID]
```

```
Command Example: Run calc.exe
Function Example: Run("calc.exe")
```

## Parameters

### Target

A document, URL, executable file (.exe, .com, .bat, etc.), shortcut (.lnk), or [system verb](#) to launch (see remarks). If *Target* is a local file and no path was specified with it, [A\\_WorkingDir](#) will be searched first. If no matching file is found there, the system will search for and launch the file if it is integrated ("known"), e.g. by being contained in one of the PATH folders.

To pass parameters, add them immediately after the program or document name. If a parameter contains spaces, it is safest to enclose it in double quotes (even though it may work without them in some cases).

### WorkingDir

The working directory for the launched item. Do not enclose the name in double quotes even if it contains spaces. If omitted, the script's own working directory (`A_WorkingDir`) will be used.

**Max|Min|Hide**

**UseErrorLevel**

If omitted, *Target* will be launched normally. Alternatively, it can contain one or more of these words:

**Max:** launch maximized

**Min:** launch minimized

**Hide:** launch hidden (cannot be used in combination with either of the above)

Note: Some applications (e.g. Calc.exe) do not obey the requested startup state and thus Max/Min/Hide will have no effect.

**UseErrorLevel:** UseErrorLevel can be specified alone or in addition to one of the above words (by separating it from the other word with a space). If the launch fails, this option skips the warning dialog, sets `ErrorLevel` to the word ERROR, and allows the `current thread` to continue. If the launch succeeds, RunWait sets `ErrorLevel` to the program's exit code, and Run sets it to 0.

When UseErrorLevel is specified, the variable `A_LastError` is set to the result of the operating system's `GetLastError()` function. `A_LastError` is a

number between 0 and 4294967295 (always formatted as decimal, not hexadecimal). Zero (0) means success, but any other number means the launch failed. Each number corresponds to a specific error condition (to get a list, search [www.microsoft.com](http://www.microsoft.com) for "system error codes"). Like `ErrorLevel`, `A_LastError` is a per-thread setting; that is, interruptions by other threads cannot change it. However, `A_LastError` is also set by `DllCall`.

### OutputVarPID

The name of the variable in which to store the newly launched program's unique `Process ID (PID)`. The variable will be made blank if the PID could not be determined, which usually happens if a system verb, document, or shortcut is launched rather than a direct executable file. `RunWait` also supports this parameter, though its `OutputVarPID` must be checked in another thread (otherwise, the PID will be invalid because the process will have terminated by the time the line following `RunWait` executes).

After the `Run` command retrieves a PID, any windows to be created by the process might not exist yet. To wait for at least one window to be created, use `winWait ahk_pid %OutputVarPID%`.

### ErrorLevel

`Run`: Does not set `ErrorLevel` unless `UseErrorLevel` (above) is in effect, in which case `ErrorLevel` is set to the word `ERROR` upon failure or `0` upon success.

RunWait: Sets ErrorLevel to the program's exit code (a signed 32-bit integer). If UseErrorLevel is in effect and the launch failed, the word ERROR is stored.

## Remarks

Unlike Run, RunWait will wait until *Target* is closed or exits, at which time ErrorLevel will be set to the program's exit code (as a signed 32-bit integer). Some programs will appear to return immediately even though they are still running; these programs spawn another process.

If *Target* contains any commas, they must be escaped as shown three times in the following example:

```
Run rundll32.exe shell32.dll,Control_RunDLL
desk.cpl,3 ; Opens Control Panel > Display
Properties > Settings
```

When running a program via Comspec (cmd.exe) -- perhaps because you need to redirect the program's input or output -- if the path or name of the executable contains spaces, the entire string should be enclosed in an outer pair of quotes. In the following example, the outer quotes are shown in red and all the inner quotes are shown in black:

```
Run %A_ComSpec% /c "'C:\My Utility.exe" "param
1" "second param" >"C:\My File.txt"
```

If *Target* cannot be launched, an error window is displayed and the current thread is exited, unless the string UseErrorLevel is included in the third

parameter or the error is caught by a [Try/Catch](#) statement.

Performance may be slightly improved if *Target* is an exact path, e.g. `Run, C:\Windows\notepad.exe "C:\My Documents\Test.txt"` rather than `Run, C:\My Documents\Test.txt`.

Special [CLSID folders](#) may be opened via Run. For example:

```
Run ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ;
Opens the "My Computer" folder.
Run ::{645ff040-5081-101b-9f08-00aa002f954e} ;
Opens the Recycle Bin.
```

System verbs correspond to actions available in a file's right-click menu in the Explorer. If a file is launched without a verb, the default verb (usually "open") for that particular file type will be used. If specified, the verb should be followed by the name of the target file. The following verbs are currently supported:

|              |                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>*verb</i> | Any system-defined or custom verb. For example: <code>Run *Compile %A_ScriptFullPath%</code><br>On Windows Vista and later, the <code>*RunAs</code> verb may be used in place of the <i>Run as administrator</i> right-click menu item.                                                                                                                                 |
| properties   | Displays the Explorer's properties window for the indicated file. For example: <code>Run, properties "C:\My File.txt"</code><br>Note: The properties window will automatically close when the script terminates. To prevent this, use <a href="#">WinWait</a> to wait for the window to appear, then use <a href="#">WinWaitClose</a> to wait for the user to close it. |
| find         | Opens an instance of the Explorer's Search Companion or Find File window at the indicated folder. For example: <code>Run, find D:\</code>                                                                                                                                                                                                                               |

|         |                                                                                                                                                                                              |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| explore | Opens an instance of Explorer at the indicated folder. For example: <code>Run, explore %A_ProgramFiles%</code> .                                                                             |
| edit    | Opens the indicated file for editing. It might not work if the indicated file's type does not have an "edit" action associated with it. For example: <code>Run, edit "C:\My File.txt"</code> |
| open    | Opens the indicated file (normally not needed because it is the default action for most file types). For example: <code>Run, open "My File.txt"</code> .                                     |
| print   | Prints the indicated file with the associated application, if any. For example: <code>Run, print "My File.txt"</code>                                                                        |

While RunWait is in a waiting state, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

## Run as Administrator

For an executable file, the *\*RunAs* verb is equivalent to selecting *Run as administrator* from the right-click menu of the file. For example, the following code attempts to restart the current script as admin:

```
full_command_line := DllCall("GetCommandLine",
"str")

if not (A_IsAdmin or
RegexMatch(full_command_line, "
/restart(?:!\S)"))
{
 try
 {
 if A_IsCompiled
 Run *RunAs "%A_ScriptFullPath%"
 }
 /restart
}
```

```
 else
 Run *RunAs "%A_AhkPath%" /restart
 "%A_ScriptFullPath%"
 }
 ExitApp
 }

 MsgBox A_IsAdmin: %A_IsAdmin%`nCommand line:
 %full_command_line%
```

If the user cancels the UAC dialog or Run fails for some other reason, the script will simply exit.

Using `/restart` ensures that a [single instance](#) prompt is not shown if the new instance of the script starts before `ExitApp` is called.

If UAC is disabled, `*RunAs` will launch the process without elevating it. Checking for `/restart` in the command line ensures that the script does not enter a runaway loop in that case. Note that `/restart` is a built-in switch, so is not included in the [array of command-line parameters](#).

The example can be modified to fit the script's needs:

- If the script absolutely requires admin rights, check `A_IsAdmin` a second time in case `*RunAs` failed to elevate the script (i.e. because UAC is disabled).
- To keep the script running even if the user cancels the UAC prompt, move `ExitApp` into the try block.
- To keep the script running even if it failed to restart (i.e. because the script file has been changed or deleted), remove `ExitApp` and use `RunWait` instead

of Run. On success, `/restart` causes the new instance to terminate the old one. On failure, the new instance exits and RunWait returns.

If UAC is enabled, the AutoHotkey installer registers the *RunAs* verb for *.ahk* files, which allows `Run *RunAs script.ahk` to launch a script as admin with the default executable.

## Related

[RunAs](#), [ProcessExist](#), [Exit](#), [CLSID List](#), [DllCall](#)

## Examples

```
Run, Notepad.exe, C:\My Documents, max

Run, mailto:someone@domain.com?subject=This is the
subject line&body=This is the message body's text.
Run, ReadMe.doc, , Max UseErrorLevel ; Launch
maximized and don't display dialog if it fails.
if ErrorLevel = "ERROR"
 MsgBox The document could not be launched.

RunWait, %A_ComSpec% /c dir c:\ >>c:\DirTest.txt,
, min
Run, c:\DirTest.txt
Run, properties c:\DirTest.txt

Run, http://www.google.com ; i.e. any URL can be
launched.
Run, mailto:someone@somedomain.com ; This should
open the default e-mail application.

Run ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ;
```

Opens the "My Computer" folder.

```
Run ::{645ff040-5081-101b-9f08-00aa002f954e} ;
```

Opens the Recycle Bin.

; To run multiple commands consecutively, use "&&" between each:

```
Run, %A_ComSpec% /c dir /b > C:\list.txt && type C:\list.txt && pause
```

; The following can be used to run a command and retrieve its output:

```
MsgBox % RunWaitOne("dir " A_ScriptDir)
```

; ...or run multiple commands in one go and retrieve their output:

```
MsgBox % RunWaitMany("
(
echo Put your commands here,
echo each one will be run,
echo and you'll get the output.
)")
```

```
RunWaitOne(command) {
```

```
 ; WshShell object:
```

```
 http://msdn.microsoft.com/en-us/library/aew9yb99
```

```
 shell := ComObjCreate("WScript.Shell")
```

```
 ; Execute a single command via cmd.exe
```

```
 exec := shell.Exec(A_ComSpec " /C " command)
```

```
 ; Read and return the command's output
```

```
 return exec.StdOut.ReadAll()
```

```
}
```

```
RunWaitMany(commands) {
```

```
 shell := ComObjCreate("WScript.Shell")
```

```
 ; Open cmd.exe with echoing of commands disabled
```

```

 exec := shell.Exec(A_ComSpec " /Q /K echo
off")
 ; Send the commands to execute, separated by
newline
 exec.StdIn.WriteLine(commands "`nexit") ;
Always exit at the end!
 ; Read and return the output of all commands
return exec.Stdout.ReadAll()
}

```

**; ExecScript: Executes the given code as a new AutoHotkey process.**

```

ExecScript(Script, Wait:=true)
{
 shell := ComObjCreate("WScript.Shell")
 exec := shell.Exec("AutoHotkey.exe
/ErrorStdOut *")
 exec.StdIn.Write(script)
 exec.StdIn.Close()
 if Wait
 return exec.Stdout.ReadAll()
}

```

**; Example:**

```

InputBox expr, Enter an expression to evaluate as
a new script.,,, Ord("*")
result := ExecScript("FileAppend `% (" expr "),
*")
MsgBox % "Result: " result

```

# RunAs

Specifies a set of user credentials to use for all subsequent uses of [Run](#) and [RunWait](#).

**RunAs** [User, Password, Domain]

**Command Example:** RunAs "User", "MyPassword", "workGroup"

**Function Example:** RunAs("User", "MyPassword", "workGroup")

## Parameters

### User

If this and the other parameters are all omitted, the RunAs feature will be turned off, which restores [Run](#) and [RunWait](#) to their default behavior. Otherwise, this is the username under which new processes will be created.

### Password

*User's password.*

### Domain

*User's domain. To use a local account, leave this blank. If that fails to work, try using @YourComputerName.*

## Remarks

If the script is running with restricted privileges due to User Account Control (UAC), any programs it launches will typically also be restricted, even if `RunAs` is used. To elevate a process, use `Run *RunAs` instead.

This command does nothing other than notify AutoHotkey to use (or not use) alternate user credentials for all subsequent uses of `Run` and `RunWait`.

`ErrorLevel` is not changed by this command. If an invalid *User*, *Password*, or *Domain* is specified, `Run` and `RunWait` will display an error message explaining the problem (unless their `UseErrorLevel` option is in effect).

While the `RunAs` feature is in effect, `Run` and `RunWait` will not be able to launch documents, URLs, or system verbs. In other words, the file to be launched must be an executable file.

The "Secondary Logon" service must be set to manual or automatic for this command to work (the OS should automatically start it upon demand if set to manual).

## Related

`Run`, `RunWait`

## Example

```
RunAs, Administrator, MyPassword
```

Run, RegEdit.exe

RunAs ; Reset to normal behavior.

# Shutdown

Shuts down, restarts, or logs off the system.

## Shutdown Code

```
Command Example: Shutdown 1
Function Example: Shutdown(1)
```

## Parameters

### Code

A combination of shutdown codes listed below.

## Remarks

The shutdown code is a combination of the following values:

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| Logoff            | 0                                                               |
| Shutdown          | 1                                                               |
| Reboot            | 2                                                               |
| Force             | 4                                                               |
| Power down        | 8                                                               |
| Suspend/Hibernate | See <a href="#">DllCall example</a> at the bottom of this page. |
| Turn monitor off  | See <a href="#">PostMessage examples</a> .                      |

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

The "Force" value (4) forces all open applications to close. It should only be used in an emergency because it may cause any open applications to lose data.

The "Power down" value shuts down the system and turns off the power.

On a related note, a script can detect when the system is shutting down or the user is logging off via [OnExit](#).

## Related

[Run](#), [ExitApp](#), [OnExit](#)

## Example

```
; Force a reboot (reboot + force = 2 + 4 = 6):
Shutdown, 6

; Call the Windows API function "SetSuspendState"
to have the system suspend or hibernate.
; Parameter #1: Pass 1 instead of 0 to hibernate
rather than suspend.
; Parameter #2: Pass 1 instead of 0 to suspend
immediately rather than asking each application
for permission.
; Parameter #3: Pass 1 instead of 0 to disable all
wake events.
DllCall("PowrProf\SetSuspendState", "int", 0,
"int", 0, "int", 0)
```

# RegDelete

Deletes a value or subkey from the registry.

**RegDelete** [KeyName, ValueName]

**RegDeleteKey** [KeyName]

**Command Example:** `RegDelete "HKEY_LOCAL_MACHINE\Software\SomeApplication", "TestValue"`

**Function Example:**  
`RegDelete("HKEY_LOCAL_MACHINE\Software\SomeApplication", "TestValue")`

## Parameters

### KeyName

The full name of the registry key.

This must start with HKEY\_LOCAL\_MACHINE, HKEY\_USERS, HKEY\_CURRENT\_USER, HKEY\_CLASSES\_ROOT, or HKEY\_CURRENT\_CONFIG (or the abbreviations for each of these, such as HKLM). To access a [remote registry](#), prepend the computer name and a slash, as in this example:

```
\\workstation01\HKEY_LOCAL_MACHINE
```

*KeyName* can be omitted only if a [registry loop](#) is running, in which case it defaults to the key of the current loop item. If the item is a subkey, the

full name of that subkey is used by default. If the item is a value, *ValueName* defaults to the name of that value, but can be overridden.

### ValueName

The name of the value to delete. If blank or omitted, the key's default value will be deleted (except as noted above). The default value is displayed as "(Default)" by RegEdit.

### ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

`A_LastError` is set to the result of the operating system's `GetLastError()` function.

### Remarks

Deleting from the registry is potentially dangerous - please exercise caution!

To retrieve and operate upon multiple registry keys or values, consider using a [registry loop](#).

Within a [registry loop](#), `RegDelete` and `RegDeleteKey` do not necessarily delete the current loop item. If the item is a subkey, `RegDelete()` only deletes its default value while `RegDeleteKey()` deletes the key itself. If the item is a value, `RegDeleteKey()` deletes the key which *contains* that value, including all subkeys and values.

For details about how to access the registry of a remote computer, see the

remarks in [registry loop](#).

To delete entries from the 64-bit sections of the registry in a 32-bit script or vice versa, use [SetRegView](#).

## Related

[RegRead](#), [RegWrite](#), [Registry-loop](#), [SetRegView](#), [IniDelete](#)

## Example

```
RegDelete,
HKEY_LOCAL_MACHINE\Software\SomeApplication,
TestValue
```

# RegRead

Reads a value from the registry.

```
OutputVar := RegRead(KeyName, ValueName)
```

```
Function Example: InstallDir :=
RegRead("HKEY_LOCAL_MACHINE\Software\AutoHotkey
", "InstallDir")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved value. If the value cannot be retrieved, the variable is made blank and [ErrorLevel](#) is set to 1.

### KeyName

The full name of the registry key.

This must start with HKEY\_LOCAL\_MACHINE, HKEY\_USERS, HKEY\_CURRENT\_USER, HKEY\_CLASSES\_ROOT, or HKEY\_CURRENT\_CONFIG (or the abbreviations for each of these, such as HKLM). To access a [remote registry](#), prepend the computer name and a slash, as in this example:

```
\\workstation01\HKEY_LOCAL_MACHINE
```

*KeyName* can be omitted only if a [registry loop](#) is running, in which case

it defaults to the key of the current loop item. If the item is a subkey, the full name of that subkey is used by default. If the item is a value, *ValueName* defaults to the name of that value, but can be overridden.

### ValueName

The name of the value to retrieve. If blank or omitted, the key's default value will be retrieved (except as noted above). The default value is displayed as "(Default)" by RegEdit. If there is no default value (that is, if RegEdit displays "value not set"), *OutputVar* is made blank and *ErrorLevel* is set to 1.

### ErrorLevel

*ErrorLevel* is set to 1 if there was a problem (such as a nonexistent key or value) or 0 otherwise.

*A\_LastError* is set to the result of the operating system's `GetLastError()` function.

### Remarks

Currently only the following value types are supported: `REG_SZ`, `REG_EXPAND_SZ`, `REG_MULTI_SZ`, `REG_DWORD`, and `REG_BINARY`.

`REG_DWORD` values are always expressed as positive decimal numbers.

When reading a `REG_BINARY` key the result is a string of hex characters. For example, the `REG_BINARY` value of `01,a9,ff,77` will be read as the string `01A9FF77`.

When reading a REG\_MULTI\_SZ key, each of the components ends in a linefeed character (`\n`). If there are no components, *OutputVar* will be made blank. See [FileSelect](#) for an example of how to extract the individual components from *OutputVar*.

To retrieve and operate upon multiple registry keys or values, consider using a [registry-loop](#).

For details about how to access the registry of a remote computer, see the remarks in [registry-loop](#).

To read and write entries from the 64-bit sections of the registry in a 32-bit script or vice versa, use [SetRegView](#).

## Related

[RegDelete](#), [RegWrite](#), [Registry-loop](#), [SetRegView](#), [IniRead](#)

## Example

```
; Example: Retrieve the path of the Program Files
directory.
```

```
; The line below ensures that the path of the 64-
bit Program Files
```

```
; directory is returned if the OS is 64-bit and
the script is not.
```

```
SetRegView 64 ; Requires v1.1.08+
```

```
RegRead, OutputVar,
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Curr
```

```
entVersion, ProgramFilesDir
MsgBox, Program files are in: %OutputVar%
```

**; Another way to retrieve the path of the Program Files directory:**

```
EnvGet OutputVar, % A_Is64bitOS ? "ProgramW6432" :
"ProgramFiles"
MsgBox, Program files are in: %OutputVar%
```

**; The following example retrieves the TYPE of a registry value (e.g. REG\_SZ or REG\_DWORD).**

```
MsgBox % RegKeyType("HKCU", "Environment", "TEMP")
return
```

**RegKeyType(RootKey, SubKey, ValueName) ; This function returns the type of the specified value.**

```
{
 Loop, Reg, %RootKey%\%SubKey%
 if (A_LoopRegName = ValueName)
 return A_LoopRegType
 return "Error"
}
```

# RegWrite

Writes a value to the registry.

```
RegWrite Value, ValueType, KeyName [, ValueName]
```

```
RegWrite Value [, ValueType, , ValueName]
```

```
Command Example: RegWrite "Test Value",
"REG_SZ",
"HKEY_LOCAL_MACHINE\SOFTWARE\TestKey",
"MyValueName"
```

```
Function Example: RegWrite("Test Value",
"REG_SZ",
"HKEY_LOCAL_MACHINE\SOFTWARE\TestKey",
"MyValueName")
```

## Parameters

### Value

The value to be written. Long text values can be broken up into several shorter lines by means of a [continuation section](#), which might improve readability and maintainability.

### ValueType

Must be either REG\_SZ, REG\_EXPAND\_SZ, REG\_MULTI\_SZ, REG\_DWORD, or REG\_BINARY.

*ValueType* can be omitted only if *KeyName* is omitted and the current

registry loop item is a value, as noted below.

### KeyName

The full name of the registry key.

This must start with HKEY\_LOCAL\_MACHINE, HKEY\_USERS, HKEY\_CURRENT\_USER, HKEY\_CLASSES\_ROOT, or HKEY\_CURRENT\_CONFIG (or the abbreviations for each of these, such as HKLM). To access a remote registry, prepend the computer name and a slash, as in this example:

```
\\workstation01\HKEY_LOCAL_MACHINE
```

*KeyName* can be omitted only if a registry loop is running, in which case it defaults to the key of the current loop item. If the item is a subkey, the full name of that subkey is used by default. If the item is a value, *ValueType* and *ValueName* default to the type and name of that value, but can be overridden.

### ValueName

The name of the value that will be written to. If blank or omitted, the key's default value will be used (except as noted above). The default value is displayed as "(Default)" by RegEdit.

### ErrorLevel

*ErrorLevel* is set to 1 if there was a problem or 0 otherwise.

*A\_LastError* is set to the result of the operating system's GetLastError() function.

## Remarks

If *KeyName* specifies a subkey which does not exist, `RegWrite` attempts to create it (along with its ancestors, if necessary). Although `RegWrite` can write directly into a root key, some operating systems might refuse to write into `HKEY_CURRENT_USER`'s top level.

If *ValueType* is `REG_DWORD`, *Value* should be between -2147483648 and 4294967295 (0xFFFFFFFF).

When writing a `REG_BINARY` key, use a string of hex characters, e.g. the `REG_BINARY` value of 01,a9,ff,77 can be written by using the string 01A9FF77.

When writing a `REG_MULTI_SZ` key, you must separate each component from the next with a linefeed character (`\n`). The last component may optionally end with a linefeed as well. No blank components are allowed. In other words, do not specify two linefeeds in a row (`\n\n`) because that will result in a shorter-than-expected value being written to the registry.

To retrieve and operate upon multiple registry keys or values, consider using a [registry-loop](#).

For details about how to access the registry of a remote computer, see the remarks in [registry-loop](#).

To read and write entries from the 64-bit sections of the registry in a 32-bit script or vice versa, use [SetRegView](#).

## Related

[RegDelete](#), [RegRead](#), [Registry-loop](#), [SetRegView](#), [IniWrite](#)

## Examples

```
RegWrite "Test Value", "REG_SZ",
"HKEY_LOCAL_MACHINE\SOFTWARE\TestKey",
"MyValueName"
RegWrite "01A9FF77", "REG_BINARY",
"HKEY_CURRENT_USER\Software\TEST_APP", "TEST_NAME"
RegWrite "Line1`nLine2", "REG_MULTI_SZ",
"HKEY_CURRENT_USER\Software\TEST_APP", "TEST_NAME"
```

# SetRegView

Sets the registry view used by RegRead, RegWrite, RegDelete and registry loops.

**SetRegView** RegView

```
Command Example: SetRegView 32
Function Example: SetRegView(32)
```

## Parameters

### RegView

Specify **32** to view the registry as a 32-bit application would, or **64** to view the registry as a 64-bit application would.

Specify the word **Default** to restore normal behaviour.

## General Remarks

This command is only useful on Windows 64-bit. It has no effect on Windows 32-bit.

On 64-bit systems, 32-bit applications run on a subsystem of Windows called **WOW64**. By default, the system redirects certain **registry keys** to prevent conflicts. For example, in a 32-bit script, `HKLM\SOFTWARE\AutoHotkey` is redirected to `HKLM\SOFTWARE\Wow6432Node\AutoHotkey`.

SetRegView allows the registry commands in a 32-bit script to access redirected keys in the 64-bit registry view and vice versa.

The built-in variable *A\_RegView* contains the current setting. Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed](#) subroutine) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[RegRead](#), [RegWrite](#), [RegDelete](#), [Loop \(registry\)](#)

## Examples

Example 1 shows how to set a specific registry view, and how registry redirection affects the script.

```
; Access the registry as a 32-bit application would.
SetRegView 32
RegWrite REG_SZ, HKLM\SOFTWARE\Test.ahk, Value,
123

; Access the registry as a 64-bit application would.
SetRegView 64
RegRead value,
HKLM\SOFTWARE\Wow6432Node\Test.ahk, Value
RegDelete HKLM\SOFTWARE\Wow6432Node\Test.ahk

MsgBox Read value '%value%' via Wow6432Node.
```

```
; Restore the registry view to the default,
which
; depends on whether the script is 32-bit or
64-bit.
SetRegView Default
; ...
```

Example 2 shows how to detect the type of EXE and operating system on which the script is running.

```
if (A_PtrSize = 8)
 script_is := "64-bit"
else ; if (A_PtrSize = 4)
 script_is := "32-bit"

if (A_Is64bitOS)
 OS_is := "64-bit"
else
 OS_is := "32-bit, which has only a single
registry view"

MsgBox("This script is %script_is%, and the OS
is %OS_is%.")
```

# ImageSearch

Searches a region of the screen for an image.

```
ImageSearch OutputVarX, OutputVarY, X1, Y1, X2, Y2,
ImageFile
```

```
Command Example: ImageSearch, x, y, 0,
0,A_ScreenWidth, A_ScreenHeight, A_ScriptDir
"\MyImage.jpg"
```

```
Function Example: ImageSearch(x, y, 0, 0,
A_ScreenWidth, A_ScreenHeight)
```

## Parameters

### OutputVarX/Y

The names of the variables in which to store the X and Y coordinates of the upper-left pixel of where the image was found on the screen (if no match is found, the variables are made blank). Coordinates are relative to the active window unless [CoordMode](#) was used to change that.

Either or both of these parameters may be left blank, in which case [ErrorLevel](#) (see below) can be used to determine whether a match was found.

### X1,Y1

The X and Y coordinates of the upper left corner of the rectangle to

search, which can be [expressions](#). **Coordinates are relative to the active window unless [CoordMode](#) was used to change that.**

## X2,Y2

The X and Y coordinates of the lower right corner of the rectangle to search, which can be [expressions](#). Coordinates are relative to the active window unless [CoordMode](#) was used to change that.

## ImageFile

The file name of an image, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified. All operating systems support GIF, JPG, BMP, ICO, CUR, and ANI images (BMP images must be 16-bit or higher). Other sources of icons include the following types of files: EXE, DLL, CPL, SCR, and other types that contain icon resources. On Windows XP or later, additional image formats such as PNG, TIF, Exif, WMF, and EMF are supported. Operating systems older than XP can be given support by copying Microsoft's free GDI+ DLL into the AutoHotkey.exe folder (but in the case of a [compiled script](#), copy the DLL into the script's folder). To download the DLL, search for the following phrase at [www.microsoft.com](http://www.microsoft.com): gdi redistributable

**Options:** Zero or more of the following strings may be also be present immediately before the name of the file. Separate each option from the next with a single space or tab. For example: `*2 *w100 *h-1`

`C:\Main Logo.bmp.`

**\*IconN:** To use an icon group other than the first one in the file, specify

`*Icon` followed immediately by the number of the group. For example, `*Icon2` would load the default icon from the second icon group.

**\*n (variation):** Specify for **n** a number between 0 and 255 (inclusive) to indicate the allowed number of shades of variation in either direction for the intensity of the red, green, and blue components of each pixel's color. For example, `*2` would allow two shades of variation. This parameter is helpful if the coloring of the image varies slightly or if *ImageFile* uses a format such as GIF or JPG that does not accurately represent an image on the screen. If you specify 255 shades of variation, all colors will match. The default is 0 shades.

**\*TransN:** This option makes it easier to find a match by specifying one color within the image that will match any color on the screen. It is most commonly used to find PNG, GIF, and TIF files that have some transparent areas (however, icons do not need this option because their transparency is automatically supported). For GIF files, `*TransWhite` might be most likely to work. For PNG and TIF files, `*TransBlack` might be best. Otherwise, specify for **N** some other color name or RGB value (see the [color chart](#) for guidance, or use [PixelGetColor](#) in its RGB mode). Examples: `*TransBlack`, `*TransFFFFFFAA`, `*Trans0xFFFFAA`.

**\*wn** and **\*hn:** Width and height to which to scale the image (this width and height also determines which icon to load from a multi-icon .ICO file). If both these options are omitted, icons loaded from ICO, DLL, or

EXE files are scaled to the system's default small-icon size, which is usually 16 by 16 (you can force the actual/internal size to be used by specifying `*w@ *h@`). Images that are not icons are loaded at their actual size. To shrink or enlarge the image while preserving its aspect ratio, specify -1 for one of the dimensions and a positive number for the other. For example, specifying `*w200 *h-1` would make the image 200 pixels wide and cause its height to be set automatically.

A [bitmap or icon handle](#) can be used instead of a filename. For example, `HBITMAP: *%handle%`.

## ErrorLevel

[ErrorLevel](#) is set to 0 if the image was found in the specified region, 1 if it was not found, or 2 if there was a problem that prevented the command from conducting the search (such as failure to open the image file or a badly formatted option).

## Remarks

ImageSearch can be used to detect graphical objects on the screen that either lack text or whose text cannot be easily retrieved. For example, it can be used to discover the position of picture buttons, icons, web page links, or game objects. Once located, such objects can be clicked via [Click](#).

A strategy that is sometimes useful is to search for a small clipping from an image rather than the entire image. This can improve reliability in cases where

the image as a whole varies, but certain parts within it are always the same. One way to extract a clipping is to:

1. Press Alt+PrintScreen while the image is visible in the active window. This places a screenshot on the clipboard.
2. Open an image processing program such as Paint.
3. Paste the contents of the clipboard (that is, the screenshot).
4. Select a region that does not vary and that is unique to the image.
5. Copy and paste that region to a new image document.
6. Save it as a small file for use with ImageSearch.

To be a match, an image on the screen must be the same size as the one loaded via the *ImageFile* parameter and its options.

The region to be searched must be visible; in other words, it is not possible to search a region of a window hidden behind another window. By contrast, images that lie partially beneath the mouse cursor can usually be detected. The exception to this is game cursors, which in most cases will obstruct any images beneath them.

Since the search starts at the top row of the region and moves downward, if there is more than one match, the one closest to the top will be found.

Icons containing a transparent color automatically allow that color to match any color on the screen. Therefore, the color of what lies behind the icon does not matter.

ImageSearch supports 8-bit color screens (256-color) or higher.

The search behavior may vary depending on the display adapter's color depth (especially for GIF and JPG files). Therefore, if a script will run under multiple color depths, it is best to test it on each depth setting. You can use the shades-of-variation option (\*n) to help make the behavior consistent across multiple color depths.

If the image on the screen is translucent, ImageSearch will probably fail to find it. To work around this, try the shades-of-variation option (\*n) or make the window temporarily opaque via `WinSetTransparent, Off`.

## Related

[PixelSearch](#), [PixelGetColor](#), [CoordMode](#), [MouseGetPos](#)

## Examples

```
ImageSearch, FoundX, FoundY, 40,40, 300, 300,
C:\My Images\test.bmp

CoordMode Pixel ; Interprets the coordinates
below as relative to the screen rather than the
active window.
ImageSearch, FoundX, FoundY, 0, 0, A_ScreenWidth,
A_ScreenHeight, *Icon3
%A_ProgramFiles%\SomeApp\SomeApp.exe
if ErrorLevel = 2
 MsgBox Could not conduct the search.
else if ErrorLevel = 1
 MsgBox Icon could not be found on the screen.
else
 MsgBox The icon was found at
```



# PixelGetColor

Retrieves the color of the pixel at the specified x,y coordinates.

```
OutputVar := PixelGetColor(X, Y [, Alt|Slow])
```

```
Function Example: color :=
PixelGetColor(100,200)
```

## Parameters

### OutputVar

The name of the variable in which to store the color ID in hexadecimal red-green-blue (RGB) format. For example, the color purple is defined 0x800080 because it has an intensity of 80 for its blue and red components but an intensity of 00 for its green component.

### X, Y

The X and Y coordinates of the pixel. Coordinates are relative to the active window unless [CoordMode](#) was used to change that.

### Alt|Slow

This parameter may contain zero or more of the following words. If more than one word is present, separate each from the next with a space (e.g. `Alt Slow`).

**Alt:** Uses an alternate method to retrieve the color, which should be used when the normal method produces invalid or inaccurate colors for a particular type of window. This method is about 10% slower than the normal method.

**Slow:** Uses a more elaborate method to retrieve the color, which may work in certain full-screen applications when the other methods fail. This method is about three times slower than the normal method. Note: *Slow* takes precedence over *Alt*, so there is no need to specify *Alt* in this case.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

The pixel must be visible; in other words, it is not possible to retrieve the pixel color of a window hidden behind another window. By contrast, pixels beneath the mouse cursor can usually be detected. The exception to this is game cursors, which in most cases will hide any pixels beneath them.

Use Window Spy (available in tray icon menu) or the example at the bottom of this page to determine the colors currently on the screen.

Known limitations:

- A window that is [partially transparent](#) or that has one of its colors marked invisible (`WinSetTransColor`) typically yields colors for the window behind

itself rather than its own.

- PixelGetColor might not produce accurate results for certain applications. If this occurs, try specifying the word *Alt* or *Slow* in the last parameter.

## Related

[PixelSearch](#), [ImageSearch](#), [CoordMode](#), [MouseGetPos](#)

## Example

```
^!z:: ; Control+Alt+Z hotkey.
MouseGetPos, mouseX, mouseY
PixelGetColor, color, %mouseX%, %mouseY%
MsgBox The color at the current cursor position is
%color%.
return
```

# PixelSearch

Searches a region of the screen for a pixel of the specified color.

```
PixelSearch OutputVarX, OutputVarY, X1, Y1, X2, Y2,
ColorID [, Variation, Fast]
```

```
Command Example: PixelSearch, x, y, 0, 0,
A_ScreenWidth, A_ScreenHeight, "0xFFFFFFFF"
Function Example: PixelSearch(x, y, 0, 0,
A_ScreenWidth, A_ScreenHeight)
```

## Parameters

### OutputVarX/Y

The names of the variables in which to store the X and Y coordinates of the first pixel that matches *ColorID* (if no match is found, the variables are made blank). Coordinates are relative to the active window unless *CoordMode* was used to change that.

Either or both of these parameters may be left blank, in which case *ErrorLevel* (see below) can be used to determine whether a match was found.

### X1, Y1

The X and Y coordinates of the upper left corner of the rectangle to search. **Coordinates are relative to the active window unless**

**CoordMode** was used to change that.

### X2, Y2

The X and Y coordinates of the lower right corner of the rectangle to search. Coordinates are relative to the active window unless **CoordMode** was used to change that.

### ColorID

The decimal or hexadecimal color ID to search for, in red-green-blue (RGB) format. For example: `0x9d6346`. Color IDs can be determined using Window Spy (accessible from the tray menu) or via **PixelGetColor**.

### Variation

A number between 0 and 255 (inclusive) to indicate the allowed number of shades of variation in either direction for the intensity of the red, green, and blue components of the color. This parameter is helpful if the color sought is not always exactly the same shade. If you specify 255 shades of variation, all colors will match. The default is 0 shades.

### Fast

This parameter contains a string of options. Currently it can only have one value (if not empty):

**Fast:** Uses a faster searching method that in most cases dramatically reduces the amount of CPU time used by the search. Although color depths as low as 8-bit (256-color) are supported, the fast mode performs much better in 24-bit or 32-bit color. If the screen's color depth is 16-bit

or lower, the *Variation* parameter might behave slightly differently in fast mode than it does in slow mode. Finally, the fast mode searches the screen row by row (top down) instead of column by column. Therefore, it might find a different pixel than that of the slow mode if there is more than one matching pixel.

## ErrorLevel

**ErrorLevel** is set to 0 if the color was found in the specified region, 1 if it was not found, or 2 if there was a problem that prevented the command from conducting the search.

## Remarks

The region to be searched must be visible; in other words, it is not possible to search a region of a window hidden behind another window. By contrast, pixels beneath the mouse cursor can usually be detected. The exception to this is cursors in games, which in most cases will hide any pixels beneath them.

Slow mode only: By default, the search starts at the upper-left pixel of the region and checks all pixels vertically beneath it for a match. If no match is found there, the search continues to the right, column by column, until it finds a matching pixel. The default left-to-right search order can be inverted by swapping *X1* and *X2* in the parameter list. In other words, if *X1* is greater than *X2*, the search will be conducted from right to left, starting at column *X1*. Similarly, if *Y1* is greater than *Y2*, each column of pixels to be searched starting at the bottom rather than the top. Finally, if the region to be searched is large and the search is repeated

with high frequency, it may consume a lot of CPU time. To alleviate this, keep the size of the area to a minimum.

## Related

[PixelGetColor](#), [ImageSearch](#), [CoordMode](#), [MouseGetPos](#)

## Example

```
PixelSearch, Px, Py, 200, 200, 300, 300, 0x9d6346,
3, Fast
if ErrorLevel
 MsgBox, That color was not found in the
 specified region.
else
 MsgBox, A color within 3 shades of variation
 was found at X%Px% Y%Py%.
```

# SoundBeep

Emits a tone from the PC speaker.

**SoundBeep** [Frequency, Duration]

```
Command Example: SoundBeep 1000, 10
Function Example: SoundBeep(1000, 10)
```

## Parameters

### Frequency

The frequency of the sound. It should be a number between 37 and 32767. If omitted, the frequency will be 523.

### Duration

The duration of the sound, in milliseconds. If omitted, the duration will be 150.

## Remarks

The script waits for the sound to finish before continuing. In addition, system responsiveness might be reduced during sound production.

If the computer lacks a sound card, a standard beep is played through the PC speaker.

To produce the standard system sounds instead of beeping the PC Speaker, see the asterisk mode of [SoundPlay](#).

## Related

[SoundPlay](#)

## Example

```
SoundBeep ; Play the default pitch and duration.
SoundBeep, 750, 500 ; Play a higher pitch for
half a second.
```

# SoundGet

Retrieves various settings from a sound device (master mute, master volume, etc.)

```
OutputVar := SoundGet(ComponentType, ControlType,
DeviceNumber)
```

```
Function Example: setting := SoundGet()
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved setting, which is either a floating point number between 0 and 100 (inclusive) or the word ON or OFF (used only for *ControlTypes* ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, and BASSBOOST). The variable will be made blank if there was a problem retrieving the setting.

### ComponentType

If omitted or blank, it defaults to the word MASTER. Otherwise, it can be one of the following words: MASTER (synonymous with SPEAKERS), DIGITAL, LINE, MICROPHONE, SYNTH, CD, TELEPHONE, PCSPEAKER, WAVE, AUX, ANALOG, HEADPHONES, or N/A. If the sound device lacks the specified *ComponentType*, *ErrorLevel* will indicate the problem.

The component labeled Auxiliary in some mixers might be accessible as ANALOG rather than AUX.

If a device has more than one instance of *ComponentType* (two of type LINE, for example), usually the first contains the playback settings and the second contains the recording settings. To access an instance other than the first, append a colon and a number to this parameter. For example: `Analog:2` is the second instance of the analog component.

### ControlType

If omitted or blank, it defaults to VOLUME. Otherwise, it can be one of the following words: VOLUME (or VOL), ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, BASSBOOST, PAN, QSOUNDPAN, BASS, TREBLE, EQUALIZER, or the number of a valid control type (see [soundcard analysis script](#)). If the specified *ComponentType* lacks the specified *ControlType*, *ErrorLevel* will indicate the problem.

**Note:** sound devices usually support only VOLUME (or VOL) and MUTE, although others may be available depending on Windows and the sound device.

### DeviceNumber

A number between 1 and the total number of supported devices. If this parameter is omitted, it defaults to 1 (the first sound device), or on Windows Vista or above, the system's default device for playback. The [soundcard analysis script](#) may help determine which number to use.

## ErrorLevel

`ErrorLevel` is set to 0 if the command succeeded. Otherwise, it is set to one of the following phrases:

- Invalid Control Type or Component Type
- Can't Open Specified Mixer
- Mixer Doesn't Support This Component Type
- Mixer Doesn't Have That Many of That Component Type
- Component Doesn't Support This Control Type
- Can't Get Current Setting

## Remarks

To discover the capabilities of the sound devices (mixers) installed on the system -- such as the available component types and control types -- run the [soundcard analysis script](#).

For more functionality and finer grained control over audio, consider using the [VA library](#).

Use [SoundSet](#) to change a setting.

## Related

[SoundSet](#), [SoundPlay](#)

## Examples



# SoundPlay

Plays a sound, video, or other supported file type.

```
SoundPlay Filename [, wait]
```

```
Command Example: SoundPlay A_ScriptDir
"\MySound.wav"
```

```
Function Example: SoundPlay(A_ScriptDir
"\MySound.wav")
```

## Parameters

### Filename

The name of the file to be played, which is assumed to be in `%A_WorkingDir%` if an absolute path isn't specified.

To produce standard system sounds, specify an asterisk followed by a number as shown below. Note: the *wait* parameter has no effect in this mode.

**\*-1:** Simple beep. If the sound card is not available, the sound is generated using the speaker.

**\*16:** Hand (stop/error)

**\*32:** Question

**\*48:** Exclamation

**\*64:** Asterisk (info)

## wait

If omitted, the script's [current thread](#) will move on to the next command(s) while the file is playing. To avoid this, specify 1 or the word WAIT, which causes the current thread to wait until the file is finished playing before continuing. Even while waiting, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

Known limitation: If the WAIT parameter is omitted, the OS might consider the playing file to be "in use" until the script closes or until another file is played (even a nonexistent file).

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

## Remarks

All Windows OSes should be able to play .wav files. However, other files (.mp3, .avi, etc.) might not be playable if the right codecs or features aren't installed on the OS.

If a file is playing and the current script plays a second file, the first file will be stopped so that the second one can play. On some systems, certain file types might stop playing even when an entirely separate script plays a new file.

To stop a file that is currently playing, use `SoundPlay` on a nonexistent filename as in this example: `SoundPlay, Nonexistent.avi`.

If the script is exited, any currently-playing file that it started will stop.

## Related

[SoundBeep](#), [SoundGet](#), [SoundSet](#), [MsgBox](#), [Threads](#)

## Example

```
SoundPlay, %A_winDir%\Media\ding.wav
SoundPlay *-1 ; Simple beep. If the sound card is
not available, the sound is generated using the
speaker.
```

# SoundSet

Changes various settings of a sound device (master mute, master volume, etc.)

```
SoundSet NewSetting [, ComponentType, ControlType,
DeviceNumber]
```

```
Command Example: SoundSet 100
Function Example: SoundSet(100)
```

## Parameters

### NewSetting

Percentage number between -100 and 100 inclusive (it can be a floating point number). If the number begins with a plus or minus sign, the **current setting** will be adjusted up or down by the indicated amount. Otherwise, the setting will be set explicitly to the level indicated by *NewSetting*.

For *ControlTypes* with only two possible settings -- namely ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, and BASSBOOST -- any positive number will turn on the setting and a zero will turn it off.

However, if the number begins with a plus or minus sign, the setting will be toggled (set to the opposite of its current state).

### ComponentType

If omitted or blank, it defaults to the word MASTER. Otherwise, it can be one of the following words: MASTER (synonymous with SPEAKERS), DIGITAL, LINE, MICROPHONE, SYNTH, CD, TELEPHONE, PCSPEAKER, WAVE, AUX, ANALOG, HEADPHONES, or N/A. If the sound device lacks the specified *ComponentType*, *ErrorLevel* will indicate the problem.

The component labeled Auxiliary in some mixers might be accessible as ANALOG rather than AUX.

If a device has more than one instance of *ComponentType* (two of type LINE, for example), usually the first contains the playback settings and the second contains the recording settings. To access an instance other than the first, append a colon and a number to this parameter. For example: `Analog:2` is the second instance of the analog component.

## ControlType

If omitted or blank, it defaults to VOLUME. Otherwise, it can be one of the following words: VOLUME (or VOL), ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, BASSBOOST, PAN, QSOUNDPAN, BASS, TREBLE, EQUALIZER, or the number of a valid control type (see [soundcard analysis script](#)). If the specified *ComponentType* lacks the specified *ControlType*, *ErrorLevel* will indicate the problem.

**Note:** sound devices usually support only VOLUME (or VOL) and MUTE, although others may be available depending on Windows and the sound device.

## DeviceNumber

A number between 1 and the total number of supported devices. If this parameter is omitted, it defaults to 1 (the first sound device), or on Windows Vista or above, the system's default device for playback. The [soundcard analysis script](#) may help determine which number to use.

## ErrorLevel

`ErrorLevel` is set to 0 if the command succeeded. Otherwise, it is set to one of the following phrases:

- Invalid Control Type or Component Type
- Can't Open Specified Mixer
- Mixer Doesn't Support This Component Type
- Mixer Doesn't Have That Many of That Component Type
- Component Doesn't Support This Control Type
- Can't Get Current Setting
- Can't Change Setting

## Remarks

For more functionality and finer grained control over audio, consider using the [VA library](#).

An alternative way to adjust the volume is to have the script send volume-control keystrokes to change the master volume for the entire system, such as in the example below:

```
Send {Volume_Up} ; Raise the master volume by
1 interval (typically 5%).
Send {Volume_Down 3} ; Lower the master volume
by 3 intervals.
Send {Volume_Mute} ; Mute/unmute the master
volume.
```

To discover the capabilities of the sound devices (mixers) installed on the system -- such as the available component types and control types -- run the [soundcard analysis script](#).

Windows 2000/XP/2003: When SoundSet changes the volume of a component, all of that component's channels (e.g. left and right) are set to the same level. In other words, any off-center "balance" that may have been set previously is lost.

Windows Vista and later: SoundSet attempts to preserve the existing balance when changing the volume level.

Use [SoundGet](#) to retrieve the current value of a setting.

## Related

[SoundGet](#), [SoundPlay](#)

## Examples

```
; BASIC EXAMPLES:
SoundSet, 50 ; Set the master volume to 50%
SoundSet +10 ; Increase master volume by 10%
SoundSet -10 ; Decrease master volume by 10%
```

```

SoundSet, 1, Microphone, mute ; mute the
microphone
SoundSet, +1, , mute ; Toggle the master mute
(set it to the opposite state)
SoundSet, +20, Master, bass ; Increase bass level
by 20%.
if ErrorLevel
 MsgBox, The BASS setting is not supported for
MASTER.

```

## Soundcard Analysis

Use the following script to discover your soundcard's capabilities (component types and control types). It displays the results in a simple ListView.

Alternatively, a script for Windows Vista and later which provides more detail (such as display names of devices) can be downloaded from the following forum topic: <http://www.autohotkey.com/board/topic/90877-/>

```

; Most of the pure numbers below probably don't
exist in any mixer, but they're queried for
completeness.
; The numbers correspond to the following items
(in order): CUSTOM, BOOLEANMETER, SIGNEDMETER,
PEAKMETER,
; UNSIGNEDMETER, BOOLEAN, BUTTON, DECIBELS,
SIGNED, UNSIGNED, PERCENT, SLIDER, FADER,
SINGLESELECT, MUX,
; MULTIPLESELECT, MIXER, MICROTIME, MILLITIME
ControlTypes :=
"VOLUME,ONOFF,MUTE,MONO,LOUDNESS,STEREOENH,BASSBOO
ST,PAN,QSOUNDPAN,BASS,TREBLE,EQUALIZER,0x00000000,
0x10010000,0x10020000,0x10020001,0x10030000,0x2001
0000,0x21010000,0x30040000,0x30020000,0x30030000,0
x30050000,0x40020000,0x50030000,0x70010000,0x70010

```

```
001,0x71010000,0x71010001,0x60030000,0x61030000"
```

```
ComponentTypes :=
"MASTER,HEADPHONES,DIGITAL,LINE,MICROPHONE,SYNTH,C
D,TELEPHONE,PCSPEAKER,WAVE,AUX,ANALOG,N/A"
```

```
; Create a ListView and prepare for the main loop:
```

```
Gui := GuiCreate()
InfoText := Gui.Add("Text", "w400 h400",
"Gathering Soundcard Info...")
LV := Gui.Add("ListView", "wp hp xp yp",
"Component Type|Control Type|Setting|Mixer")
LV.Visible := false
LV.ModifyCol(4, "Integer")
Gui.Show()
```

```
Loop ; For each mixer number that exists in the
system, query its capabilities.
```

```
{
 CurrMixer := A_Index
 Setting := SoundGet(,, CurrMixer)
 if ErrorLevel = "Can't Open Specified Mixer" ;
Any error other than this indicates that the mixer
exists.
```

```
 break
```

```
 ; For each component type that exists in this
mixer, query its instances and control types:
```

```
 For i, CurrComponent in StrSplit(ComponentTypes,
 ",")
 {
```

```
 ; First check if this component type even
exists in the mixer:
```

```
 Setting := SoundGet(CurrComponent,, CurrMixer)
 if ErrorLevel = "Mixer Doesn't Support This
Component Type"
```

```
 continue ; Start a new iteration to move on
to the next component type.
```

```

Loop ; For each instance of this component
type, query its control types.
{
 CurrInstance := A_Index
 ; First check if this instance of this
instance even exists in the mixer:
 Setting := SoundGet(CurrComponent ":"
CurrInstance,, CurrMixer)
 ; Checking for both of the following errors
allows this script to run on older versions:
 if ErrorLevel = "Mixer Doesn't Have That
Many of That Component Type"
 || ErrorLevel = "Invalid Control Type or
Component Type"
 break ; No more instances of this
component type.
 ; Get the current setting of each control
type that exists in this instance of this
component:
 For i, CurrControl in StrSplit(ControlTypes,
",")
 {
 Setting := SoundGet(CurrComponent ":"
CurrInstance, CurrControl, CurrMixer)
 ; Checking for both of the following
errors allows this script to run on older
versions:
 if ErrorLevel = "Component Doesn't Support
This Control Type"
 || ErrorLevel = "Invalid Control Type or
Component Type"
 continue
 if ErrorLevel ; Some other error, which
is unexpected so show it in the results.
 Setting := ErrorLevel
 ComponentString := CurrComponent
 if CurrInstance > 1

```

```
 ComponentString := ":" CurrInstance
 if Setting is "float"
 Setting := Format("{:0.2f}", Setting) ;
Limit number of decimal places in percentages to
two.
 LV.Add(, ComponentString, CurrControl,
Setting, CurrMixer)
 } ; For each control type.
} ; For each component instance.
} ; For each component type.
} ; For each mixer.

Loop(LV.GetCount("Col")) ; Auto-size each column
to fit its contents.
 LV.ModifyCol(A_Index, "AutoHdr")

InfoText.Visible := false
LV.Visible := true
return
```

# Chr()

Returns the string (usually a single character) corresponding to the character code indicated by the specified number.

```
String := Chr(Number)
```

## Parameters

### Number

If Unicode is supported, *Number* is a Unicode character code between 0 and 0x10FFFF; otherwise it is an ANSI character code between 0 and 255.

## Return Value

This function returns a string corresponding to *Number*. If *Number* is not in the valid range of character codes, an empty string is returned.

## Remarks

The meaning of character codes greater than 127 depends on the [string encoding](#) in use, which in turn depends on whether a [Unicode or ANSI](#) executable is in use.

Common character codes include 9 (tab), 10 (linefeed), 13 (carriage return), 32 (space), 48-57 (the digits 0-9), 65-90 (uppercase A-Z), and 97-122 (lowercase a-

z).

## Related

[Ord](#)

## Examples

```
MsgBox, % Chr(116) ; Shows "t".
```

# Format

Formats a variable number of input values according to a format string.

```
OutputVar := Format(Format, Value1 [, Value2, Value3
...])
```

```
Function Example: hex := format("0x{1:X}", 100)
```

## Parameters

### OutputVar

The name of the variable in which to store the formatted string.

### FormatStr

A format string composed of literal text and placeholders of the form

`{Index}Format`.

*Index* is an integer indicating which input value to use, where 1 is the first value.

*Format* is an optional format specifier, as described below.

Omit the index to use the next input value in the sequence (even if it has been used earlier in the string). For example, `"{2:i} { :i}"` formats the second and third input values as decimal integers, separated by a space. If *Index* is omitted, *Format* must still be preceded by `:`. Specify

empty braces to use the next input value with default formatting: `{ }`

Use `{ }` and `{ }` to include literal braces in the string. Any other invalid placeholders are included in the result as is.

Whitespace inside the braces is not permitted (except as a flag).

## Values

Input values to be formatted and inserted into the final string. Each value is a separate parameter. The first value has an index of 1.

To pass an array of values, use a [variadic function call](#):

```
arr := [13, 240]
MsgBox % Format("{2:x}{1:02x}", arr*)
```

## Format Specifiers

Each format specifier can include the following components, in this order (without the spaces):

Flags Width .Precision ULT Type

**Flags** which affect output justification and prefixes: `-` `+` `0` space `#`

**Width:** a decimal integer which controls the minimum width of the formatted value, in characters. By default, values are right-aligned and spaces are used for padding. This can be overridden by using the `-` (left-align) and `0` (zero prefix)

flags.

**.Precision:** a decimal integer which controls the maximum number of string characters, decimal places, or significant digits to output, depending on the output type. It must be preceded by a decimal point. Specifying a precision may cause the value to be truncated or rounded.

- **f, e, E:** *Precision* specifies the number of digits after the decimal point. The default is 6.
- **g, G:** *Precision* specifies the maximum number of significant digits. The default is 6.
- **S:** *Precision* specifies the maximum number of characters to be printed. Characters in excess of this are not printed.
- For the integer types (**d, i, u, x, X, o**), *Precision* acts like *Width* with the **0** prefix and a default of 1.

**ULT:** specifies a case transformation to apply to a string value -- Upper, Lower or Title. Valid only with the **S** type. For example **{:U}** or **{:.20Ts}**. Lowercase **l** and **L** are also supported, but **U** is reserved for unsigned integers.

**Type:** a character indicating how the input value should be interpreted. If omitted, it defaults to **S**.

| Flag     | Meaning                                                                                    | Default      |
|----------|--------------------------------------------------------------------------------------------|--------------|
| <b>-</b> | Left align the result within the given field width (insert spaces to the right if needed). | Right align. |
|          |                                                                                            | Sign appears |

|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| +     | Use a sign (+ or -) to prefix the output value if it is of a signed type.                                                                                                                                                                                                                                                                                                                                                                                          | only for negative signed values (-). |
| 0     | If <i>width</i> is prefixed by 0, leading zeros are added until the minimum width is reached. If both 0 and - appear, the 0 is ignored. If 0 is specified as an integer format (i, u, x, X, o, d) and a precision specification is also present - for example, { :04.d } - the 0 is ignored.                                                                                                                                                                       | No padding.                          |
| space | Use a single space to prefix the output value with a space if it is signed and positive. The space is ignored if both the space and + flags appear.                                                                                                                                                                                                                                                                                                                | No space appears.                    |
| #     | <p>When it's used with the o, x, or X format, the # flag uses 0, 0x, or 0X, respectively, to prefix any nonzero output value.</p> <p>When it's used with the e, E, f, a or A format, the # flag forces the output value to contain a decimal point.</p> <p>When it's used with the g or G format, the # flag forces the output value to contain a decimal point and prevents the truncation of trailing zeros.</p> <p>Ignored when used with c, d, i, u, or s.</p> |                                      |

| Type Character | Argument | Output format                                                                                                     |
|----------------|----------|-------------------------------------------------------------------------------------------------------------------|
| d or i         | Integer  | Signed decimal integer.                                                                                           |
| u              | Integer  | Unsigned decimal integer.                                                                                         |
|                |          | Unsigned hexadecimal integer; uses "abcdef" or "ABCDEF" depending on the case of x. The 0x prefix is not included |

|                                  |                |                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code> or <code>X</code> | Integer        | unless the <code>#</code> flag is used, as in <code>{:#x}</code> . To always include the prefix, use <code>0x{ :x}</code> or similar.                                                                                                                                                                                                                                                                |
| <code>o</code>                   | Integer        | Unsigned octal integer.                                                                                                                                                                                                                                                                                                                                                                              |
| <code>f</code>                   | Floating-point | Signed value that has the form <code>[ - ]ddd.dddd</code> , where <code>ddd</code> is one or more decimal digits. The number of digits before the decimal point depends on the magnitude of the number, and the number of digits after the decimal point depends on the requested precision.                                                                                                         |
| <code>e</code>                   | Floating-point | Signed value that has the form <code>[ - ]d.dddd e [sign]dd[d]</code> where <code>d</code> is one decimal digit, <code>ddd</code> is one or more decimal digits, <code>dd[d]</code> is two or three decimal digits depending on the output format and size of the exponent, and <code>sign</code> is <code>+</code> or <code>-</code> .                                                              |
| <code>E</code>                   | Floating-point | Identical to the <code>e</code> format except that <code>E</code> rather than <code>e</code> introduces the exponent.                                                                                                                                                                                                                                                                                |
| <code>g</code>                   | Floating-point | Signed values are displayed in <code>f</code> or <code>e</code> format, whichever is more compact for the given value and precision. The <code>e</code> format is used only when the exponent of the value is less than <code>-4</code> or greater than or equal to the <i>precision</i> argument. Trailing zeros are truncated, and the decimal point appears only if one or more digits follow it. |
| <code>G</code>                   | Floating-point | Identical to the <code>g</code> format, except that <code>E</code> , rather than <code>e</code> , introduces the exponent (where appropriate).                                                                                                                                                                                                                                                       |
|                                  |                | Signed hexadecimal double-precision floating-point value that has the form <code>[?]0xh.hhhh p±dd</code> , where <code>h.hhhh</code> are the hex digits (using lower case letters) of the                                                                                                                                                                                                            |

|                |                |                                                                                                                                                                  |
|----------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>a</code> | Floating-point | mantissa, and <i>dd</i> are one or more digits for the exponent. The precision specifies the number of digits after the point.                                   |
| <code>A</code> | Floating-point | Identical to the <code>a</code> format, except that <i>P</i> , rather than <i>p</i> , introduces the exponent.                                                   |
| <code>p</code> | Integer        | Displays the argument as a memory address in hexadecimal digits.                                                                                                 |
| <code>s</code> | String         | Specifies a string. If the input value is numeric, it is automatically converted to a string before the <i>Width</i> and <i>Precision</i> arguments are applied. |
| <code>c</code> | Character code | Specifies a single character by its ordinal value, similar to <code>Chr(n)</code> . If the input value is outside the expected range, it wraps around.           |

## Remarks

Unlike `printf`, size specifiers are not supported. All integers and floating-point input values are 64-bit.

## Related

[FormatTime](#)

## Example

```
; Simple substitution
s := Format("{2}, {1}!`r`n", "World", "Hello")
; Padding with spaces
```

```
s .= Format("|{: -10}|`r`n|{:10}|`r`n", "Left",
"Right")
```

```
; Hexadecimal
```

```
s .= Format("{1:#x} {2:X} 0x{3:x}`r`n",
3735928559, 195948557, 0)
```

```
; Floating-point
```

```
s .= Format("{1:0.3f} {1:.10f}", 4*atan(1))
```

```
ListVars ; Use AutoHotkey's main window to
display monospaced text.
```

```
WinWaitActive ahk_class AutoHotkey
```

```
ControlSetText(s, "Edit1")
```

```
WinWaitClose
```

# FormatTime

Transforms a `YYYYMMDDHH24MISS` timestamp into the specified date/time format.

```
OutputVar := FormatTime(YYYYMMDDHH24MISS, Format)
```

```
Function Example: now :=
FormatTime(A_Now, "hh:mm")
```

## Parameters

### OutputVar

The name of the variable in which to store the result.

### YYYYMMDD...

Leave this parameter blank to use the current local date and time.

Otherwise, specify all or the leading part of a timestamp in the `YYYYMMDDHH24MISS` format. If the date and/or time portion of the timestamp is invalid -- such as February 29th of a non-leap year -- the date and/or time will be omitted from *OutputVar*. Although only years between 1601 and 9999 are supported, a formatted time can still be produced for earlier years as long as the time portion is valid.

### Format

If omitted, it defaults to the time followed by the long date, both of which

will be formatted according to the current user's locale. For example: 4:55 PM Saturday, November 27, 2004

Otherwise, specify one or more of the date-time formats below, along with any literal spaces and punctuation in between (commas do not need to be escaped; they can be used normally). In the following example, note that M must be capitalized: M/d/yyyy h:mm tt

## Date Formats (case sensitive)

|      |                                                                                    |
|------|------------------------------------------------------------------------------------|
| d    | Day of the month without leading zero (1 - 31)                                     |
| dd   | Day of the month with leading zero (01 – 31)                                       |
| ddd  | Abbreviated name for the day of the week (e.g. Mon) in the current user's language |
| dddd | Full name for the day of the week (e.g. Monday) in the current user's language     |
| M    | Month without leading zero (1 – 12)                                                |
| MM   | Month with leading zero (01 – 12)                                                  |
| MMM  | Abbreviated month name (e.g. Jan) in the current user's language                   |
| MMMM | Full month name (e.g. January) in the current user's language                      |
| y    | Year without century, without leading zero (0 – 99)                                |
| yy   | Year without century, with leading zero (00 - 99)                                  |
| yyyy | Year with century. For example: 2005                                               |
| gg   | Period/era string for the current user's locale (blank if none)                    |

## Time Formats (case sensitive)

|  |  |
|--|--|
|  |  |
|--|--|

|    |                                                                   |
|----|-------------------------------------------------------------------|
| h  | Hours without leading zero; 12-hour format (1 - 12)               |
| hh | Hours with leading zero; 12-hour format (01 - 12)                 |
| H  | Hours without leading zero; 24-hour format (0 - 23)               |
| HH | Hours with leading zero; 24-hour format (00- 23)                  |
| m  | Minutes without leading zero (0 - 59)                             |
| mm | Minutes with leading zero (00 - 59)                               |
| s  | Seconds without leading zero (0 - 59)                             |
| ss | Seconds with leading zero (00 - 59)                               |
| t  | Single character time marker, such as A or P (depends on locale)  |
| tt | Multi-character time marker, such as AM or PM (depends on locale) |

**The following formats must be used **alone**; that is, with no other formats or text present in the *Format* parameter. These formats are not case sensitive.**

|           |                                                                                                                                                              |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (Blank)   | Leave <i>Format</i> blank to produce the time followed by the long date. For example, in some locales it might appear as 4:55 PM Saturday, November 27, 2004 |
| Time      | Time representation for the current user's locale, such as 5:26 PM                                                                                           |
| ShortDate | Short date representation for the current user's locale, such as 02/29/04                                                                                    |
| LongDate  | Long date representation for the current user's locale, such as Friday, April 23, 2004                                                                       |
| YearMonth | Year and month format for the current user's locale, such as February, 2004                                                                                  |
| YDay      | Day of the year without leading zeros (1 - 366)                                                                                                              |

|       |                                                                                                                                                                                                                                                                                                                                          |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YDay0 | Day of the year with leading zeros (001 – 366)                                                                                                                                                                                                                                                                                           |
| WDay  | Day of the week (1 – 7). Sunday is 1.                                                                                                                                                                                                                                                                                                    |
| YWeek | The ISO 8601 full year and week number. For example: 200453. If the week containing January 1st has four or more days in the new year, it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. Consequently, both January 4th and the first Thursday of January are always in week 1. |

## Additional Options

The following options can appear inside the `YYYYMMDDHH24MISS` parameter immediately after the timestamp (if there is no timestamp, they may be used alone). In the following example, note the lack of commas between the last four items:

```
FormatTime, OutputVar, 20040228 LSys D1 D4
```

**R:** Reverse. Have the date come before the time (meaningful only when *Format* is blank).

**Ln:** If this option is *not* present, the current user's locale is used to format the string. To use the system's locale instead, specify `LSys`. To use a specific locale, specify the letter `L` followed by a hexadecimal or decimal locale identifier (LCID). For information on how to construct an LCID, search [www.microsoft.com](http://www.microsoft.com) for the following phrase: Locale Identifiers

**Dn:** Date options. Specify for **n** one of the following numbers:

0: Force the default options to be used. This also causes the short date to be in

effect.

1: Use short date (meaningful only when *Format* is blank; not compatible with 2 and 8).

2: Use long date (meaningful only when *Format* is blank; not compatible with 1 and 8).

4: Use alternate calendar (if any).

8: Use Year-Month format (meaningful only when *Format* is blank; not compatible with 1 and 2).

0x10: Add marks for left-to-right reading order layout.

0x20: Add marks for right-to-left reading order layout.

0x80000000: Do not obey any overrides the user may have in effect for the system's default date format.

0x40000000: Use the system ANSI code page for string translation instead of the locale's code page.

**Tn**: Time options. Specify for **n** one of the following numbers:

0: Force the default options to be used. This also causes minutes and seconds to be shown.

1: Omit minutes and seconds.

2: Omit seconds.

4: Omit time marker (e.g. AM/PM).

8: Always use 24-hour time rather than 12-hour time.

12: Combination of the above two.

0x80000000: Do not obey any overrides the user may have in effect for the system's default time format.

0x40000000: Use the system ANSI code page for string translation instead of the

locale's code page.

**Note:** Dn and Tn may be repeated to put more than one option into effect, such as this example: `FormatTime, OutputVar, 20040228 D2 D4 T1 T8`

## Remarks

Letters and numbers that you want to be transcribed literally from *Format* into *OutputVar* should be enclosed in single quotes as in this example: `'Date: MM/dd/yy 'Time:' hh:mm:ss tt.`

By contrast, non-alphanumeric characters such as spaces, tabs, linefeeds (`^n`), slashes, colons, commas, and other punctuation do not need to be enclosed in single quotes. The exception to this is the single quote character itself: to produce it literally, use four consecutive single quotes (`''''`), or just two if the quote is already inside an outer pair of quotes.

If *Format* contains date and time elements together, they must not be intermixed. In other words, the string should be dividable into two halves: a time half and a date half. For example, a format string consisting of `"hh yyyy mm"` would not produce the expected result because it has a date element in between two time elements.

When *Format* contains a numeric day of the month (either `d` or `dd`) followed by the full month name (MMMM), the genitive form of the month name is used (if the language has a genitive form).

If *Format* contains more than 2000 characters, *OutputVar* will be made blank.

On a related note, addition and subtraction of dates and times can be performed with [DateAdd](#) and [DateDiff](#).

## Related

To convert in the reverse direction -- that is, *from* a formatted date/time *to* [YYYYMMDDHH24MISS](#) format -- see [www.autohotkey.com/forum/topic20405.html](http://www.autohotkey.com/forum/topic20405.html)

See also: [Gui DateTime control](#), [built-in date and time variables](#), [FileGetTime](#)

## Examples

```
FormatTime, TimeString
MsgBox The current time and date (time first) is
%TimeString%.
```

```
FormatTime, TimeString, R
MsgBox The current time and date (date first) is
%TimeString%.
```

```
FormatTime, TimeString,, Time
MsgBox The current time is %TimeString%.
```

```
FormatTime, TimeString, T12, Time
MsgBox The current 24-hour time is %TimeString%.
```

```
FormatTime, TimeString,, LongDate
MsgBox The current date (long format) is
%TimeString%.
```

```
FormatTime, TimeString, 20050423220133, dddd MMMM
d, yyyy hh:mm:ss tt
MsgBox The specified date and time`, when
formatted`, is %TimeString%.

FormatTime, TimeString, 200504, 'Month Name':
MMMM`n'Day Name': dddd
MsgBox %TimeString%

FormatTime, YearWeek, 20050101, YWeek
MsgBox January 1st of 2005 is in the following ISO
year and week number: %YearWeek%
```

```
; Change the date-time stamp of a file:
FileSelect, FileName, 3,, Pick a file
if FileName = "" ; The user didn't pick a file.
 return
FileGetTime, FileTime, %FileName%
FormatTime, FileTime, %FileTime% ; Since the
last parameter is omitted, the long date and time
are retrieved.
MsgBox The selected file was last modified at
%FileTime%.
```

```
; The following function converts the specified
number of seconds into the corresponding
; number of hours, minutes, and seconds (hh:mm:ss
format).

MsgBox % FormatSeconds(7384) ; 7384 = 2 hours + 3
minutes + 4 seconds. It yields: 2:03:04

FormatSeconds(NumberOfSeconds) ; Convert the
specified number of seconds to hh:mm:ss format.
```

```
{
 time := 19990101 ; *Midnight* of an arbitrary
date.
 time := DateAdd(time, NumberOfSeconds,
"Seconds")
 FormatTime, mmss, %time%, mm:ss
 return NumberOfSeconds//3600 ":" mmss
/*
; Unlike the method used above, this would not
support more than 24 hours worth of seconds:
 FormatTime, hmmss, %time%, h:mm:ss
 return hmmss
*/
}
```

# InStr()

Searches for a given occurrence of a string, from the left or the right.

```
FoundPos := InStr(Haystack, Needle [, CaseSensitive
:= false, StartingPos := 1, Occurrence := 1])
```

## Parameters

### Haystack

The string whose content is searched.

### Needle

The string to search for.

### CaseSensitive

If the parameter *CaseSensitive* is omitted or false, the search is not case sensitive (the method of insensitivity depends on [StringCaseSense](#)); otherwise, the case must match exactly.

### StartingPos

If *StartingPos* is omitted, it defaults to 1 (the beginning of *Haystack*). Otherwise, specify 2 to start at the second character, 3 to start at the third, and so on.

If *StartingPos* is negative, the search is conducted in reverse (right-to-left), starting at that position from the right. For example, -1 starts at the

last character. If *StartingPos* is 0 or beyond the length of *Haystack*, 0 is returned.

Regardless of the value of *StartingPos*, the return value is always relative to the first character of *Haystack*. For example, the position of "abc" in "123abc789" is always 4.

### Occurrence

If *Occurrence* is omitted, it defaults to the first match of the *Needle* in *Haystack*. Specify 2 for *Occurrence* to return the position of the second match, 3 for the third match, etc.

## Return Value

This function returns the position of an occurrence of the string *Needle* in the string *Haystack*. Position 1 is the first character; this is because 0 is synonymous with "false", making it an intuitive "not found" indicator.

## Remarks

[RegExMatch](#) can be used to search for a pattern (regular expression) within a string, making it much more flexible than `InStr()`. However, `InStr()` is generally faster than `RegExMatch()` when searching for a simple substring.

`InStr()` searches only up to the first binary zero (null-terminator), whereas `RegExMatch()` searches the entire [length](#) of the string even if it includes binary zero.

## Related

[RegexMatch](#), [StringCaseSense](#), *Value is Type*

## Example

```
; Example 1
```

```
MsgBox % InStr("123abc789","abc") ; Returns 4
```

```
; Example 2
```

```
Haystack := "The Quick Brown Fox Jumps Over the
Lazy Dog"
```

```
Needle := "Fox"
```

```
If InStr(Haystack, Needle)
```

```
 MsgBox, The string was found.
```

```
Else
```

```
 MsgBox, The string was not found.
```

```
; Example 3
```

```
Haystack := "The Quick Brown Fox Jumps Over the
Lazy Dog"
```

```
Needle := "the"
```

```
MsgBox % InStr(Haystack, Needle, false, 1, 2) ;
```

```
case insensitive search, return start position of
second occurrence
```

```
MsgBox % InStr(Haystack, Needle, true) ; case
```

```
sensitive search, return start position of first
occurrence, same result as above
```

# Ord()

Returns the ordinal value (numeric character code) of the first character in the specified string.

```
Number := Ord(String)
```

## Parameters

### String

The string whose ordinal value is retrieved.

## Return Value

This function returns the ordinal value of *String*, or 0 if *String* is empty. If *String* begins with a Unicode supplementary character, this function returns the corresponding Unicode character code (a number between 0x10000 and 0x10FFFF). Otherwise it returns a value in the range 0 to 255 (for ANSI) or 0 to 0xFFFF (for Unicode). See [Unicode vs ANSI](#) for details.

## Related

[Chr](#)

## Examples

```
; Both message boxes below show 116, because only
```

the first character is considered.

```
MsgBox, % Ord("t")
```

```
MsgBox, % Ord("test")
```

# RegExMatch()

Determines whether a string contains a pattern (regular expression).

```
FoundPos := RegExMatch(Haystack, NeedleRegEx [, OutputVar, StartingPosition := 1])
```

## Parameters

### Haystack

The string whose content is searched. This may contain binary zero.

### NeedleRegEx

The pattern to search for, which is a Perl-compatible regular expression (PCRE). The pattern's *options* (if any) must be included at the beginning of the string followed by a close-parenthesis. For example, the pattern `"i)abc.*123"` would turn on the case-insensitive option and search for "abc", followed by zero or more occurrences of any character, followed by "123". If there are no options, the ")" is optional; for example, `")abc"` is equivalent to `"abc"`.

Although *NeedleRegEx* cannot contain binary zero, the pattern `\x00` can be used to match a binary zero within *Haystack*.

### OutputVar

*OutputVar* is the unquoted name of a variable in which to store a *match*

`object`, which can be used to retrieve the position, length and value of the overall match and of each `captured subpattern`, if any are present.

If the pattern is not found (that is, if the function returns 0), this variable is made blank.

### StartingPosition

If *StartingPosition* is omitted, it defaults to 1 (the beginning of *Haystack*). Otherwise, specify 2 to start at the second character, 3 to start at the third, and so on. If *StartingPosition* is beyond the length of *Haystack*, the search starts at the empty string that lies at the end of *Haystack* (which typically results in no match).

Specify a negative *StartingPosition* to start at that position from the right. For example, -1 starts at the last character and -2 starts at the next-to-last character. If *StartingPosition* tries to go beyond the left end of *Haystack*, all of *Haystack* is searched.

Regardless of the value of *StartingPosition*, the return value is always relative to the first character of *Haystack*. For example, the position of "abc" in "123abc789" is always 4.

### Return Value

This function returns the position of the leftmost occurrence of *NeedleRegEx* in the string *Haystack*. Position 1 is the first character. Zero is returned if the pattern is not found.

## Errors

**Syntax errors:** If the pattern contains a syntax error, an [exception](#) is thrown with a message in the following form: *Compile error N at offset M: description*. In that string, *N* is the PCRE error number, *M* is the position of the offending character inside the regular expression, and *description* is the text describing the error.

**Execution errors:** If an error occurs during the *execution* of the regular expression, an [exception](#) is thrown. The *Extra* property of the exception object contains the PCRE error number. Although such errors are rare, the ones most likely to occur are "too many possible empty-string matches" (-22), "recursion too deep" (-21), and "reached match limit" (-8). If these happen, try to redesign the pattern to be more restrictive, such as replacing each `*` with a `?`, `+`, or a limit like `{0,3}` wherever feasible.

## Options

See [Options](#) for modifiers such as `"i)abc"`, which turns off case-sensitivity in the pattern `"abc"`.

## Match Object

If a match is found, an object containing information about the match is stored in *OutputVar*. This object has the following properties:

**Match.Pos(N):** Returns the position of the overall match or a captured

subpattern.

**Match.Len(N)**: Returns the length of the overall match or a captured subpattern.

**Match.Value(N)**: Returns the overall match or a captured subpattern.

**Match.Name(N)**: Returns the name of the given subpattern, if it has one.

**Match.Count()**: Returns the overall number of subpatterns.

**Match.Mark()**: Returns the *NAME* of the last encountered `(*MARK:NAME)`, when applicable.

**Match[N]**: If *N* is 0 or a valid subpattern number or name, this is equivalent to `Match.Value(N)`. Otherwise, *N* can be the name of one of the above properties. For example, `Match["Pos"]` and `Match.Pos` are equivalent to `Match.Pos()` unless a subpattern named "Pos" exists, in which case they are equivalent to `Match.Value("Pos")`.

**Match.N**: Same as above, except that *N* is an unquoted name or number.

For all of the above properties, *N* can be any of the following:

- 0 for the overall match.
- The number of a subpattern, even one that also has a name.
- The name of a subpattern.

Brackets [] may be used in place of parentheses () if *N* is specified.

The object does not support enumeration; that is, the [for-loop](#) is not supported.

Instead, use `Loop Match.Count()`.

## Performance

To search for a simple substring inside a larger string, use `InStr` because it is faster than `RegexMatch()`.

To improve performance, the 100 most recently used regular expressions are kept cached in memory (in compiled form).

The `study option (S)` can sometimes improve the performance of a regular expression that is used many times (such as in a loop).

## Remarks

A subpattern may be given a name such as the word *Year* in the pattern "(?P<Year>\d{4})". Such names may consist of up to 32 alphanumeric characters and underscores. Note that named subpatterns are also numbered, so if an unnamed subpattern occurs after "Year", it would be stored in `OutputVar[2]`, not `OutputVar[1]`.

Most characters like `abc123` can be used literally inside a regular expression. However, the characters `\.*?+[\{\}\^\$` must be preceded by a backslash to be seen as literal. For example, `\.` is a literal period and `\\` is a literal backslash. Escaping can be avoided by using `\Q...\E`. For example: `\QLiteral Text\E`.

Within a regular expression, special characters such as tab and newline can be escaped with either an accent (`^`) or a backslash (`\`). For example, `^t` is the same as

\t except when the **x** option is used.

To learn the basics of regular expressions (or refresh your memory of pattern syntax), see the [RegEx Quick Reference](#).

AutoHotkey's regular expressions are implemented using Perl-compatible Regular Expressions (PCRE) from [www.pcre.org](http://www.pcre.org).

## Related

[RegExReplace](#), [RegEx Quick Reference](#), [Regular Expression Callouts](#), [InStr](#), [SubStr](#), [SetTitleMatchMode RegEx](#), [Global matching and Grep \(forum link\)](#)

Common sources of text data: [FileRead](#), [Download](#), [Clipboard](#), [GUI Edit controls](#)

## Examples

```
FoundPos := RegExMatch("xxxabc123xyz", "abc.*xyz")
; Returns 4, which is the position where the match
was found.
```

```
FoundPos := RegExMatch("abc123123", "123$") ;
Returns 7 because the $ requires the match to be
at the end.
```

```
FoundPos := RegExMatch("abc123", "i)^ABC") ;
Returns 1 because a match was achieved via the
case-insensitive option.
```

```
FoundPos := RegExMatch("abcXYZ123", "abc(.*)123",
SubPat) ; Returns 1 and stores "XYZ" in
SubPat[1].
```

```
FoundPos := RegExMatch("abc123abc456", "abc\d+",
"", 2) ; Returns 7 instead of 1 due to
```

StartingPosition 2 vs. 1.

; For general RegEx examples, see the [RegEx Quick Reference](#).

# RegexReplace()

Replaces occurrences of a pattern (regular expression) inside a string.

```
NewStr := RegexReplace(Haystack, NeedleRegex [, Replacement := "", OutputVarCount := "", Limit := -1, StartingPosition := 1])
```

## Parameters

### Haystack

The string whose content is searched and replaced. This may contain binary zero.

### NeedleRegex

The pattern to search for, which is a Perl-compatible regular expression (PCRE). The pattern's [options](#) (if any) must be included at the beginning of the string followed by a close-parenthesis. For example, the pattern `"i)abc.*123"` would turn on the case-insensitive option and search for "abc", followed by zero or more occurrences of any character, followed by "123". If there are no options, the ")" is optional; for example, `")abc"` is equivalent to `"abc"`.

Although *NeedleRegex* cannot contain binary zero, the pattern `\x00` can be used to match a binary zero within *Haystack*.

### Replacement

The string to be substituted for each match, which is plain text (not a regular expression). It may include backreferences like `$1`, which brings in the substring from *Haystack* that matched the first *subpattern*. The simplest backreferences are `$0` through `$9`, where `$0` is the substring that matched the entire pattern, `$1` is the substring that matched the first subpattern, `$2` is the second, and so on. For backreferences above 9 (and optionally those below 9), enclose the number in braces; e.g.  `${10}`,  `${11}`, and so on. For *named subpatterns*, enclose the name in braces; e.g.  `${SubpatternName}`. To specify a literal `$`, use  `$$` (this is the only character that needs such special treatment; backslashes are never needed to escape anything).

To convert the case of a subpattern, follow the `$` with one of the following characters: `U` or `u` (uppercase), `L` or `l` (lowercase), `T` or `t` (title case, in which the first letter of each word is capitalized but all others are made lowercase). For example, both  `$U1` and  `$U{1}` transcribe an uppercase version of the first subpattern.

Nonexistent backreferences and those that did not match anything in *Haystack* -- such as one of the subpatterns in `"(abc)|(xyz)"` -- are transcribed as empty strings.

### OutputVarCount

The unquoted name of a variable in which to store the number of replacements that occurred (0 if none).

### Limit

If *Limit* is omitted, it defaults to -1, which replaces **all** occurrences of the pattern found in *Haystack*. Otherwise, specify the maximum number of replacements to allow. The part of *Haystack* to the right of the last replacement is left unchanged.

### StartingPosition

If *StartingPosition* is omitted, it defaults to 1 (the beginning of *Haystack*). Otherwise, specify 2 to start at the second character, 3 to start at the third, and so on. If *StartingPosition* is beyond the length of *Haystack*, the search starts at the empty string that lies at the end of *Haystack* (which typically results in no replacements).

Specify a negative *StartingPosition* to start at that position from the right. For example, -1 starts at the last character and -2 starts at the next-to-last character. If *StartingPosition* tries to go beyond the left end of *Haystack*, all of *Haystack* is searched.

Regardless of the value of *StartingPosition*, the return value is always a complete copy of *Haystack* -- the only difference is that more of its left side might be unaltered compared to what would have happened with a *StartingPosition* of 1.

### Return Value

This function returns a version of *Haystack* whose contents have been replaced by the operation. If no replacements are needed, *Haystack* is returned unaltered.

## Errors

An [exception](#) is thrown if:

- the pattern contains a syntax error; or
- an error occurred during the *execution* of the regular expression.

For details, see [RegexMatch](#).

## Options

See [Options](#) for modifiers such as "[i](#)abc", which turns off case-sensitivity in the pattern "abc".

## Performance

To replace simple substrings, use [StrReplace](#) because it is faster than `RegexReplace()`.

If you know what the maximum number of replacements will be, specifying that for the *Limit* parameter improves performance because the search can be stopped early (this might also reduce the memory load on the system during the operation). For example, if you know there can be only one match near the beginning of a large string, specify a limit of 1.

To improve performance, the 100 most recently used regular expressions are kept cached in memory (in compiled form).

The [study option \(S\)](#) can sometimes improve the performance of a regular

expression that is used many times (such as in a loop).

## Remarks

Most characters like abc123 can be used literally inside a regular expression. However, the characters `\.*?+[{|^$` must be preceded by a backslash to be seen as literal. For example, `\.` is a literal period and `\\` is a literal backslash. Escaping can be avoided by using `\Q...\E`. For example: `\QLiteral Text\E`.

Within a regular expression, special characters such as tab and newline can be escaped with either an accent (```) or a backslash (`\`). For example, ``t` is the same as `\t`.

To learn the basics of regular expressions (or refresh your memory of pattern syntax), see the [RegEx Quick Reference](#).

## Related

[RegExMatch](#), [RegEx Quick Reference](#), [Regular Expression Callouts](#), [StrReplace](#), [InStr](#)

Common sources of text data: [FileRead](#), [Download](#), [Clipboard](#), [GUI Edit controls](#)

## Examples

```
NewStr := RegExReplace("abc123123", "123$", "xyz")
; Returns "abc123xyz" because the $ allows a match
only at the end.
```

```
NewStr := RegExReplace("abc123", "i)^ABC") ;
Returns "123" because a match was achieved via the
case-insensitive option.
```

```
NewStr := RegExReplace("abcXYZ123", "abc(.*)123",
"aaa$1zzz") ; Returns "aaaXYZzzz" by means of the
$1 backreference.
```

```
NewStr := RegExReplace("abc123abc456", "abc\d+",
"", ReplacementCount) ; Returns "" and stores 2
in ReplacementCount.
```

```
; For general RegEx examples, see the RegEx Quick
Reference.
```

# Sort

Arranges a variable's contents in alphabetical, numerical, or random order (optionally removing duplicates).

```
OutputVar := Sort(String [, Options])
```

## Parameters

### OutputVar

The name of the variable in which to store the result.

### String

The string to sort.

### Options

See list below.

## Options

A string of zero or more of the following letters (in any order, with optional spaces in between):

**C**: Case sensitive sort (ignored if the **N** option is also present). If both **C** and **CL** are omitted, the uppercase letters A-Z are considered identical to their lowercase counterparts for the purpose of the sort.

**CL:** Case insensitive sort based on the current user's locale. For example, most English and Western European locales treat the letters A-Z and ANSI letters like Ä and Ü as identical to their lowercase counterparts. This method also uses a "word sort", which treats hyphens and apostrophes in such a way that words like "coop" and "co-op" stay together. Depending on the content of the items being sorted, the performance will be 1 to 8 times worse than the default method of insensitivity.

**Dx:** Specifies *x* as the delimiter character, which determines where each item begins and ends. If this option is not present, *x* defaults to linefeed (`\n`), which correctly sorts the string if its lines end in either LF (`\n`) or CR+LF (`\r\n`).

**F MyFunction:** Uses custom sorting according to the criteria in *MyFunction* (though sorting takes much longer). Specify the letter "F" followed by optional spaces/tabs followed by the name of a [function](#) to be used for comparing any two items in the list. The function must accept two or three parameters. When the function deems the first parameter to be greater than the second, it should return a positive integer; when it deems the two parameters to be equal, it should return 0, "", or nothing; otherwise, it should return a negative integer. If a decimal point is present in the returned value, that part is ignored (i.e. 0.8 is the same as 0). If present, the third parameter receives the offset (in characters) of the second item from the first as seen in the original/unordered list (see examples). Finally, the function uses the same global settings (e.g. [StringCaseSense](#)) as the Sort command that called it.

Note: the **F** option causes all other options except **D**, **Z**, and **U** to be ignored (though **N**, **C**, and **CL** still affect how [duplicates](#) are detected). Also, sorting

does not occur when the specified function: 1) does not exist; 2) accepts fewer than two parameters; or 3) the first or second parameter is [ByRef](#).

**N:** Numeric sort: Each item is assumed to be a number rather than a string (for example, if this option is not present, the string 233 is considered to be less than the string 40 due to alphabetical ordering). Both decimal and hexadecimal strings (e.g. 0xF1) are considered to be numeric. Strings that do not start with a number are considered to be zero for the purpose of the sort. Numbers are treated as 64-bit floating point values so that the decimal portion of each number (if any) is taken into account.

**Pn:** Sorts items based on character position **n** (do not use hexadecimal for **n**). If this option is not present, **n** defaults to 1, which is the position of the first character. The sort compares each string to the others starting at its **n**th character. If **n** is greater than the length of any string, that string is considered to be blank for the purpose of the sort. When used with option **N** (numeric sort), the string's character position is used, which is not necessarily the same as the number's digit position.

**R:** Sorts in reverse order (alphabetically or numerically depending on the other options).

**Random:** Sorts in random order. This option causes all other options except **D**, **Z**, and **U** to be ignored (though **N**, **C**, and **CL** still affect how duplicates are detected). Examples:

```
Sort, MyVar, %MyVar%, Random
Sort, MyVar, %MyVar%, Random Z D |
```

---

**U:** Removes duplicate items from the list so that every item is unique.

**ErrorLevel** is set to the number of items removed (0 if none). If the **C** option is in effect, the case of items must match for them to be considered identical. If the **N** option is in effect, an item such as 2 would be considered a duplicate of 2.0. If either the **Pn** or \ (backslash) option is in effect, the entire item must be a duplicate, not just the substring that is used for sorting. If the **Random** or **F/Function** option is in effect, duplicates are removed only if they appear adjacent to each other as a result of the sort. For example, when "A|B|A" is sorted randomly, the result could contain either one or two A's.

**Z:** To understand this option, consider a variable that contains RED`nGREEN`nBLUE`n. If the **Z** option is not present, the last linefeed (`n) is considered to be part of the last item, and thus there are only 3 items. But by specifying **Z**, the last `n (if present) will be considered to delimit a blank item at the end of the list, and thus there are 4 items (the last being blank).

**\:** Sorts items based on the substring that follows the last backslash in each. If an item has no backslash, the entire item is used as the substring. This option is useful for sorting bare filenames (i.e. excluding their paths), such as the example below, in which the AAA.txt line is sorted above the BBB.txt line because their directories are ignored for the purpose of the sort:

```
C:\BBB\AAA.txt
C:\AAA\BBB.txt
```

Note: Options **N** and **P** are ignored when the backslash option is present.

## Remarks

This command is typically used to sort a variable that contains a list of lines, with each line ending in a linefeed character (`\n`). One way to get a list of lines into a variable is to load an entire file via [FileRead](#).

`ErrorLevel` is changed by this command only when the `U` option is in effect.

If a large variable was sorted and later its contents are no longer needed, you can free its memory by making it blank, e.g. `MyVar := ""`.

## Related

[FileRead](#), [file-reading loop](#), [parsing loop](#), [StrSplit](#), [RegisterCallback](#), [Clipboard](#)

## Examples

```
MyVar := "5,3,7,9,1,13,999,-4"
Sort MyVar, %MyVar%, N D, ; Sort numerically, use
comma as delimiter.
MsgBox %MyVar% ; The result is
-4,1,3,5,7,9,13,999

; The following example sorts the contents of a
file:
FileRead Contents, C:\Address List.txt
if not ErrorLevel ; Successfully loaded.
{
 Sort Contents, %Contents%
 FileDelete, C:\Address List (alphabetical).txt
 FileAppend, %Contents%, C:\Address List
(alphabetical).txt
```

```
Contents := "" ; Free the memory.
}
```

```
; The following example makes Win+C a hotkey to
copy files from an open
; Explorer window and put their sorted filenames
onto the clipboard:
```

```
#c:
Clipboard := "" ; Must be blank for detection to
work.
```

```
Send ^c
```

```
ClipWait 2
```

```
if ErrorLevel
```

```
return
```

```
Sort Clipboard, %Clipboard%
```

```
MsgBox Ready to be pasted:`n%Clipboard%
```

```
return
```

```
; The following examples demonstrate custom
sorting via a callback function.
```

```
MyVar := "def`nabc`nmno`nFGH`nco-
op`ncoop`ncop`ncon`n"
```

```
Sort, MyVar, %MyVar%, F StringSort
```

```
StringSort(a1, a2)
```

```
{
```

```
return a1 > a2 ? 1 : a1 < a2 ? -1 : 0 ; Sorts
```

```
alphabetically based on the setting of
```

```
StringCaseSense.
```

```
}
```

```
MyVar := "5,3,7,9,1,13,999,-4"
```

```
Sort, MyVar, %MyVar%, F IntegerSort D,
```

```
IntegerSort(a1, a2)
```

```
{
```

```
return a1 - a2 ; Sorts in ascending numeric
```

```
order. This method works only if the difference
is never so large as to overflow a signed 64-bit
```

```
integer.
```

```
}
```

```
MyVar := "1,2,3,4"
```

```
Sort, MyVar, %MyVar%, F ReverseDirection D, ;
```

```
Reverses the list so that it contains 4,3,2,1
```

```
ReverseDirection(a1, a2, offset)
```

```
{
```

```
 return offset ; Offset is positive if a2 came
after a1 in the original list; negative otherwise.
```

```
}
```

# StringCaseSense

Determines whether string comparisons are case sensitive (default is "not case sensitive").

**StringCaseSense** On|Off|Locale

```
Command Example: StringCaseSense On
Function Example: StringCaseSense("On")
```

## Parameters

### On|Off|Locale

**On:** String comparisons are case sensitive. This setting also makes the `expression equal sign operator (=)` and the case-insensitive mode of `InStr` use the *locale* method described below.

**Off** (starting default): The letters A-Z are considered identical to their lowercase counterparts. This is the starting default for all scripts due to backward compatibility and performance (*Locale* is 1 to 8 times slower than *Off* depending on the nature of the strings being compared).

**Locale** String comparisons are case **insensitive** according to the rules of the current user's locale. For example, most English and Western European locales treat not only the letters A-Z as identical to their lowercase counterparts, but also ANSI letters like Ä and Ü as identical to

theirs.

## Remarks

This setting applies to:

- Expression comparison operators (except `==`). However, since the equal-sign operator (`=`) is always case-insensitive, it uses the *Locale* mode when *StringCaseSense* is *On*, as does the case-insensitive mode of `InStr`.
- `InStr`. However, it is not affected when its *CaseSensitive* parameter is *true*.
- `StrReplace`

The built-in variable `A_StringCaseSense` contains the current setting (the word *On*, *Off*, or *Locale*).

Every newly launched *thread* (such as a *hotkey*, *custom menu item*, or *timed subroutine*) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

`StrReplace`, `InStr`

## Example

```
StringCaseSense Locale
```

# StrLower / StrUpper

Converts a string to lowercase or uppercase.

```
OutputVar := StrLower(String [, T])
```

```
Function Example: lower := StrLower("HELLO")
```

## Parameters

### OutputVar

The name of the variable in which to store the newly converted string.

### String

The string to convert.

### T

If this parameter is the letter T, the string will be converted to title case. For example, "GONE with the WIND" would become "Gone With The Wind".

## Remarks

To detect whether a character or string is entirely uppercase or lowercase, use *value* is "upper"/"lower" or `RegexMatch`. For example:

```
var := "abc"
```

```
if var is "upper"
 MsgBox var is empty or contains only
uppercase characters.
if var is "lower"
 MsgBox var is empty or contains only
lowercase characters.
if RegExMatch(var, "[a-z]+$")
 MsgBox var is not empty and contains only
lowercase ASCII characters.
if !RegExMatch(var, "[A-Z]")
 MsgBox var does not contain any uppercase
ASCII characters.
```

Format can also be used for case conversions, as shown below:

```
MsgBox % Format("{:U}, {:L} and {:T}", "upper",
"LOWER", "title")
```

## Related

[InStr](#), [SubStr](#), [StrLen](#), [StrReplace](#)

## Example

```
StrUpper, String1, %String1% ; i.e. output can be
the same as input.
StrLower, String2, ABC ; String2 now contains
"abc"
```

# StrReplace

Replaces the specified substring with a new string.

```
OutputVar := StrReplace(Haystack, SearchText ,
ReplaceText, OutputVarCount, Limit = -1)
```

```
Function Example: string := StrReplace(text,
100, 200)
```

## Parameters

### OutputVar

The name of the variable in which to store the result of the replacement process.

### Haystack

The string whose content is searched and replaced.

### SearchText

The string to search for. Matching is not case sensitive unless [StringCaseSense](#) has been turned on.

### ReplaceText

*SearchText* will be replaced with this text. If omitted or blank, *SearchText* will be replaced with blank (empty). In other words, it will be omitted

from *OutputVar*.

### OutputVarCount

The unquoted name of a variable in which to store the number of replacements that occurred (0 if none).

### Limit

If *Limit* is omitted, it defaults to -1, which replaces **all** occurrences of the pattern found in *Haystack*. Otherwise, specify the maximum number of replacements to allow. The part of *Haystack* to the right of the last replacement is left unchanged.

## Remarks

The built-in variables `%A_Space%` and `%A_Tab%` contain a single space and a single tab character, respectively. They are useful when searching for spaces and tabs alone or at the beginning or end of *SearchText*.

## Related

[RegExReplace](#), [StringCaseSense](#), [InStr](#), [SubStr](#), [StrLen](#), [StrLower](#), [StrUpper](#)

## Examples

```
; Remove all CR+LF's from the clipboard contents:
```

```
StrReplace, clipboard, %clipboard%, r n
```

```
; Replace all spaces with pluses:
```

```
StrReplace, NewStr, %OldStr%, %A_SPACE%, +
```

```
; Remove all blank lines from the text in a
variable:
```

```
Loop
{
 StrReplace, MyString, %MyString%, `r`n`r`n,
 `r`n, ErrorLevel
 if ErrorLevel = 0 ; No more replacements
needed.
 break
}
```

# StrSplit

Separates a string into an [array](#) of substrings using the specified delimiters.

```
Array := StrSplit(String [, Delimiters, OmitChars])
```

## Parameters

### String

A string to split.

### Delimiters

If this parameter is blank or omitted, each character of the input string will be treated as a separate substring.

Otherwise, *Delimiters* can be either a single string or an array of strings, each of which is used to determine where the boundaries between substrings occur. Since the delimiters are not considered to be part of the substrings themselves, they are never included in the returned array. Also, if there is nothing between a pair of delimiters within the input string, the corresponding array element will be blank.

For example: `","` would divide the string based on every occurrence of a comma. Similarly, `[A_Tab, A_Space]` would create a new array element every time a space or tab is encountered in the input string.

### OmitChars

An optional list of characters (case sensitive) to exclude from the beginning and end of each array element. For example, if *OmitChars* is `"t"`, spaces and tabs will be removed from the beginning and end (but not the middle) of every element.

If *Delimiters* is blank, *OmitChars* indicates which characters should be excluded from the array.

## Remarks

Whitespace characters such as spaces and tabs will be preserved unless those characters are included in the *Delimiters* or *OmitChars* parameters. Tabs and spaces can be trimmed from both ends of any variable by using [Trim](#). For example: `var := Trim(var)`

To split a string that is in standard CSV (comma separated value) format, use a [parsing loop](#) since it has built-in CSV handling.

To arrange the fields in a different order prior to splitting them, use the [Sort](#) command.

If you do not need the substrings to be permanently stored in memory, consider using a [parsing loop](#) -- especially if the input string is very large, in which case a large amount of memory would be saved. For example:

```
Colors := "red,green,blue"
Loop, parse, %Colors%, `,
 MsgBox Color number %A_Index% is
%A_LoopField%.
```

---

## Related

[Parsing loop](#), [Arrays](#), [Sort](#), [SplitPath](#), [InStr](#), [SubStr](#), [StrLen](#), [StrLower](#), [StrUpper](#), [StrReplace](#)

## Examples

```
TestString := "This is a test."
word_array := StrSplit(TestString, A_Space, ".")
; Omits periods.
MsgBox("The 4th word is " word_array[4])

Colors := "red,green,blue"
ColorArray := StrSplit(Colors, ",")
Loop ColorArray.Length()
{
 this_color := ColorArray[a_index]
 MsgBox, Color number %a_index% is
 %this_color%.
}
```

# SubStr()

Retrieves one or more characters from the specified position in a string.

```
NewStr := SubStr(String, StartingPos [, Length])
```

## Parameters

### String

The string whose content is copied. This may contain binary zero.

### StartingPos

Specify 1 to start at the first character, 2 to start at the second, and so on (if *StartingPos* is 0 or beyond *String*'s length, an empty string is returned).

Specify a negative *StartingPos* to start at that position from the right. For example, -1 extracts the last character and -2 extracts the two last characters (but if *StartingPos* tries to go beyond the left end of the string, the extraction starts at the first character).

### Length

If this parameter is omitted, it defaults to "all characters". Otherwise, specify the maximum number of characters to retrieve (fewer than the maximum are retrieved whenever the remaining part of the string is too short). You can also specify a negative *Length* to omit that many characters from the end of the returned string (an empty string is returned

if all or too many characters are omitted).

## Return Value

This function returns the requested substring of *String*.

## Related

[RegExMatch](#)

## Examples

```
; Example 1
```

```
MsgBox % SubStr("123abc789", 4, 3) ; Returns abc
```

```
; Example 2
```

```
String := "The Quick Brown Fox Jumps Over the Lazy
Dog"
```

```
MsgBox % SubStr(String, 1, 19) ; Returns "The
Quick Brown Fox"
```

```
MsgBox % SubStr(String, -8) ; Returns "Lazy Dog"
```

# Trim

Trims characters from the beginning and/or end of a string.

```
OutputVar := Trim(String, OmitChars)
OutputVar := LTrim(String, OmitChars)
OutputVar := RTrim(String, OmitChars)
```

```
Function Example: result := Trim(text, "`n`r")
```

## Parameters

### OutputVar

The name of the variable in which to store the new string.

### String

Any string value or variable. Numbers are not supported.

### OmitChars

An optional list of characters (case sensitive) to exclude from the beginning and/or end of *String*. If omitted, spaces and tabs will be removed.

## Examples

```
text := " text "
MsgBox % "No trim:`t "` text ""
```



# Struct

Struct() is a build-in function that creates and returns a special structure object. This special object can be used to access the structure its fields using object syntax.

```
OutputVar := Struct(Definition [,AddressToStructure, InitObject])
```

```
Function Example: pt := Struct("int x;y", {x:10,y:20})
```

## Parameters

### OutputVar

The name of the variable in which to store the Structure Object.

### Definition

This parameter must be a string containing the structure definition.

Definition is similar to C so most structures can be transformed very easily, also [default data types](#) can be used.

Either semicolon (;) or comma (,) can be used to separate fields, even mixed.

If no type for first field is given UInt will be used, otherwise previous type will be used, e.g. in "**a,Int x,y,Char c,d**", a will be UInt, y Int and d

Char.

If only one type/key is given, e.g. "UInt", it is assumed to be a structure definition that resolves to default type or a variable that describes the structure.

**Note!** something like "len;" or "MyVar," are interpreted as "UInt length" and "UInt MyVar" unless len and MyVar are a variable defining the structure.

So for **Struct("MyVar")**, MyVar must define a structure, e.g. **MyVar := "Int x, Int y"**, if there is no MyVar variable or it is empty **Struct("MyVar")** will be same as **Struct("UInt MyVar")**.

```
_POINT:="(
Int x; // also comments in C style are
supported
 // empty lines will be simply
ignored.
Int y; // last ; is optional and can be
omitted.
)"
```

New lines can be omitted too. This way the definition can be written much more compact.

```
_POINT:="Int x,Int y"
```

UInt is default type so it can be omitted too if negative values are not relevant (for negative values Int must be used).

```
_POINT:="x,y"
```

## UNION AND STRUCT

Struct supports unions and structures in structures, note such structures do not have a name so you can't use same name for a field in main structure and sub-structure.

```
_MyUnion:="
(
union {
 UInt int;
 struct {
 UShort x;
 UShort y;
 };
 struct {
 Byte a;
 Byte b;
 Byte c;
 Byte d;
 };
};
)"
mys:=Struct(_MyUnion)
mys.int:=0xFFFFFFFF
MsgBox % mys.int "`n" mys.x " " mys.y "`n"
mys.a " " mys.b " " mys.c " " mys.d
```

Create structure from a string containing the structure definition.

```
pt:=Struct("UInt x,UInt y")
```

The definition can be saved in a variable. Struct will also resolve the given string to the variable if necessary.

```
POINT:="UInt x,UInt y"
pt:=Struct("POINT")
```

An existing structure object can be used too.

```
POINT:=Struct("x,y")
pt:= Struct(POINT)
```

### Global / Static / Local variables

Inside a function you can also use local and static variables for structure definition.

Even when this objects are returned the static variable reference in definition can be resolved dynamically.

It is also possible to create a structure from a static variable even if you are not inside the function. Therefore include the name of the function and enclose variable in brackets. This is also necessary if you create a static structure from a static variable like in MyFunc here.

```
MyFunc() ; using this method we can create
static structures
AnotherFunc() ; this method can be used
anywhere to address the variable
pt:=Struct("MyFunc(POINT)",{x:100,y:200}) ;
even outside the function we can access the
definition variable
MsgBox % pt.x "-" pt.y
MyFunc(){
```

```

static POINT:="UInt x,UInt y"
 , pt:=Struct("MyFunc(POINT)",
{x:10,y:20})
 MsgBox(pt.x "-" pt.y)
}
AnotherFunc(){
 static pt:=Struct("MyFunc(POINT)",
{x:10,y:20})
 MsgBox(pt.x "-" pt.y)
}

```

### AddressToStructure

Address of memory representing the structure. This variable is used to access an existing structure in memory.

When no AddressToStructure is given but InitObject is used, it must be passed in this parameter, see InitObject example. For example here we would use the memory of variable **pointMem** for the structure.

```

VarSetCapacity(pointMem,8)
pt:=Struct("x,y",&pointMem)
pt.x:=10
MsgBox % pt[] " = " (&pointMem) "`n" pt.x "
= " NumGet(pointMem,"UInt")

```

### InitObject

Initialize your structure right away. To initialize the structure you will need to pass an object with keys and values.

The order of keys and values is not relevant because it will be enumerated in alphabetical order anyway.

When no AddressToStructure is given, InitObject must be used in

AddressToStructure parameter.

```
pt:=Struct("x,y",{x:100,y:200})
MsgBox % pt.x "`n" pt.y
```

## Methods

### Size

Returns the size in bytes of a structure or its field.

```
OutputVar := Struct.Size([field])
```

**OutputVar** The name of the Variable in which to store the size of field or structure.  
**field** Name of existing field within the structure, if omitted the size of the structure is returned.

```
pt:=Struct("x,y")
Msgbox % pt.Size() ; returns 8
```

```
struct:=Struct("Int64 x,y")
Msgbox % struct.Size("y") ; returns 8
```

If structure is an array you will need to pass a digit to retrieve the size of a field.

```
struct:=Struct("Int64[2]")
Msgbox % struct.Size(1) ; returns 8
```

### Offset

Returns offset for a field.

```
OutputVar := Struct.Offset(field)
```

**OutputVar** The name of the Variable in which to store the offset.

field Name of existing field within the structure.

```
pt:=Struct("x,y")
MsgBox % pt.Offset("y") ; returns 4
```

## CountOf

Returns size of array or 0 if structure or field is not an array.

```
OutputVar := Struct.CountOf([field])
```

**OutputVar** The name of the Variable in which to store the length of array.

field Name of existing field within the structure..

```
uint:=Struct("UInt[10]")
MsgBox % uint.CountOf() ; returns 10
pt:=Struct("UInt x[2],UInt y[2]")
MsgBox % pt.CountOf("x") ; returns 2
```

## Fill

Fills the structure with given value.

## Struct.Fill([value])

**value** A digit value or character that will fill the structure. Mainly used for structures.

```
pt:=Struct("x,y",{x:10,y:20})
MsgBox % pt.x
pt.Fill() ; same as pt.Fill(0)
MsgBox % pt.x
s:=Struct("Byte[10]")
s.Fill("A")
MsgBox % StrGet(s[],10,"CP0")
```

## GetAddress

Returns address of field or structure.

```
OutputVar := Struct.GetAddress([field])
```

**OutputVar** The name of the Variable in which to store the address.

**field** Name of existing field within the structure. When omitted, returns the structure itself. To get the address for structure you can also use structure objects and [""] for its fields. **Note** you cannot use [] for

```
pt:=Struct("x,y")
MsgBox % pt.GetAddress() " = " pt[]
MsgBox % pt.GetAddress("x") " = " pt.x[""]
```

## Encoding

Returns encoding for field or structure.

```
OutputVar := Struct.Encoding([field])
```

**OutputVar** The name of the Variable in which to store the encoding.

field Name of existing field within the structure. When omitted, Encoding itself will be returned.

If type of field or structure is not one of String types (TCHAR, CHAR, UCHAR, LPTSTR...) -1 is returned. Otherwise it returns 0 for CP0 and 1200 for UTF-16 ...

Other encoding types have to use StrGet and StrSet to retrieve correct text.

```
str1:=Struct("LPTSTR name")
str2:=Struct("LPTSTR")
MsgBox % str1.Encoding("name") " = "
str2.Encoding()
```

## IsPointer

Returns true if the field or structure is a pointer.

```
OutputVar := Struct.IsPointer([field])
```

**OutputVar** The name of the Variable in which to store true if field or structure is a pointer / false otherwise.

field Name of existing field within the structure. When omitted, return itself is a pointer.

```
s:=Struct("UInt *a,UInt b")
MsgBox % s.IsPointer("a") "`n"
s.IsPointer("b")
s:=Struct("UInt*")
MsgBox % s.IsPointer()
```

## GetPointer

Returns pointer saved in structure or field.

```
OutputVar := Struct.GetPointer([field])
```

**OutputVar** The name of the Variable in which to store the address.

field Name of existing field within the structure. When omitted, read pointer of first array item in our structure.

```
str:=Struct("LPTSTR name",{name:"AutoHotkey"})
MsgBox % str.GetPointer("name") "`n"
StrGet(str.GetPointer("name"))
```

You can also use "" to read the pointer. So ["" ] returns address and ["" ,"" ] returns the pointer, ["" ,"" ,"" ] pointer to pointer and so on.

```
str:=Struct("LPTSTR name",{name:"AutoHotkey"})
MsgBox % str.name["",""] "`n"
StrGet(str.name["",""])
```

## SetCapacity

Set new size for our structure or pointer, returns true if new memory was allocated.

```
OutputVar := Struct.SetCapacity([field,] newsiz)
```

**OutputVar** The name of the Variable in which to store the new size.

**field** Name of the field where capacity should be set, if omitted structure capacity will be set. This is mainly used for arrays to shrink or increase the size. If capacity for main structure, its old memory will be freed if necessary and new memory will be zero-filled.

**new size** Must be a digit or a variable containing a digit that represents new size of field or structure.

```
str:=Struct("LPTSTR name")
str.SetCapacity("name",2000)
previous_pointer := str.GetPointer("name")
str.name:="AutoHotkey"
MsgBox % previous_pointer " = "
str.GetPointer("name") "`n" str.name ; as you
can see pointer did not change.
```

You can allocate and use memory as you like, for example here we will store a String and the pointer in same memory block. Of course when you allocate new memory here you will loose contents.

```
str:=Struct("LPTSTR")
```

```
str.SetCapacity(2000)
str.1[""]:=str[]+A_PtrSize
str.1:="AutoHotkey"
MsgBox % str.1
```

## GetCapacity

Returns Capacity previously set using .SetCapacity() or via assigning a string.

```
OutputVar := Struct.GetCapacity([field])
```

**OutputVar** The name of the Variable in which to store the capacity in bytes  
field Name of existing field in our structure, if omitted returns structure capacity

```
str:=Struct("LPTSTR name")
str.SetCapacity("name",2000)
MsgBox % str.GetCapacity("name")
```

## Clone

Returns new structure object of same type.

```
OutputVar :=
Struct.Clone([AddressOfStructure,InitObject])
```

**OutputVar** The name of the Variable in which to store the new structure  
AddressOfStructure Address to memory for this structure. When omitted, memory is allocated

reserved and set internally.

InitObject

An object used to initialize the structure.

```
pt:=Struct("x,y")
pt1:=pt.Clone({x:10,y:20})
MsgBox % pt1.x "-" pt1.y
```

You can do the same using the new operator

```
pt:=Struct("x,y")
pt1:= new pt({x:10,y:20})
MsgBox pt1.x "-" pt1.y
```

## Features and Remarks

Some remarks about structure objects and more features.

A structure object cannot be altered, so you cannot add more fields to it. The only exception are arrays, here items are resolved dynamically. You can receive the address of structure or key using empty key (e.g. struct.item[""]). For structure objects also [] can be used (e.g. struct[] or struct[""]).

When a key is not given a type, e.g. "LPTSTR key1,key2", previous

type is used. If the first key lacks a type, UInt is used, so "**key1,key2**" is equivalent to "**UInt key1,key2**" or "**UInt key1,UInt key2**".

Note: pointer needs to be specified for each element, so "**\*UInt key1,key2**" is equivalent to "**UInt \*key1,UInt key2**". If both elements are pointers "**UInt \*key1,\*key2**" must be used.

To access a pointer in pointer you can specify empty key several times, e.g. object["", ""] would get the pointer at address of object[]. Same is valid for keys, so here you'll receive the pointer.

```
s:=Struct("LPTSTR str")
s.str:="Hello World!"
MsgBox % StrGet(s.str["", ""])
```

**TYPE ONLY DEFINITION** Struct supports type only definition for all default types like Int,Byte,Char... .

To access fields of such structures you will allways need to use digits like in arrays.

```
u:=Struct("UInt") ; this is similar to
UInt[1]
u.1:=10
MsgBox % u.1
```

**ARRAYS** Same way arrays are supported.

```
u:=Struct("UInt[10]")
```

```
u.10:=100
MsgBox % u.10
```

**ARRAYS OF UNKNOWN SIZE** Any structure can be used as array. However be careful, accessing memory that does not belong to the structure may crash your application.

```
pt:=Struct("x,y")
pt.SetCapacity(16) ; increase capacity to
hold 2 structures
pt.x := 1 ; we can access the first field
right away, similar to pt.1.x := 1
pt.2.x := 2 ; here we access the second
structure
MsgBox % pt.2.x " != " pt.x
```

**ENUMERATING A STRUCTURE** Using a for loop we can enumerate the structure to retrieve field names and their values. Enumeration will be executed in same order as the structure was defined, not alphabetically like for simple objects.

```
s:=Struct("Byte x,Int u,LPTSTR str")
s.x:=10
s.u:=1000
s.str:="AutoHotkey"
for k,v in s
 MsgBox % k ": " v
```

Also dedicated Arrays can be enumerated same way.

```
x:=Struct("UInt[10]", [0,9,8,7,6,5,4,3,2,1])
for k, v In x
```

```
MsgBox % k " : " v
```

**DYNAMIC AND STATIC FIELDS AND STRUCTURES** Struct supports calling dynamic structures and fields, such that are defined but do not exist as an object/field.

These objects/fields are created dynamically whenever needed and invoked/called automatically internally.

A static field on the other hand is defined directly in structure object. For Example, here x and y are static fields.

```
myStruct:=Struct("Int x, Int y")
```

Dynamic fields and structures are little different to static, they use reference to the definition to create the structure internally.

So here x and y fields will be created dynamically only when necessary.

```
POINT:="Int x, Int y"
pt:=Struct("POINT p", { p: { x:10, y:20 } }
) ; same structure as above but POINT
structure is resoved dynamically.
MsgBox % pt.p.x " , " pt.p.y
```

Furthermore, you can allocate memory for static fields using .SetCapacity() method, see below. This memory will be managed internally and freed whenever the object is deleted (last reference is released).

This feature allows assigning a string without initializing memory manually before.

```
s:=Struct("LPTSTR string")
s.string:="Hello World!"
MsgBox % s.string
```

The size of allocated memory is backed up internally and can be retrieved with `.GetCapacity()` method

```
MsgBox % s.GetCapacity("string")
```

Whenever a new string is assigned, memory will be only reallocated if a larger memory is needed.

To free the memory manually use `.SetCapacity()` method.

```
s.SetCapacity("string",0)
MsgBox % s.GetCapacity("string")
```

You can manually allocate memory to fields using `.SetCapacity()` method.

```
s:=Struct("LPTSTR string")
s.SetCapacity("string",260)
```

**NOTE** You cannot allocate memory for dynamic fields.

Calling `.GetCapacity()` method on dynamic field will return -1.

Also calling `.SetCapacity()` method will simply fail.

## BIT FIELDS

Also bit fields are supported, see [Bit Fields](#) for more information.

```
Bits:=Struct("
```

```

(
 {
 Byte int;
 struct {
 Byte a:1,b:1,c:1,d:1,e:1,f:1,g:1,h:1;
 }
 }
)");
Loop % 0xFF{
 bit:=(Bits.int:=A_Index) "`t"
 For k, v in Bits
 If A_Index>1
 bit.= v " "
 ToolTip % "int bits: 1 2 3 4 5 6 7 8`n"
 bit
 Sleep 100
}

```

## Related

[sizeof](#), [StructTypes](#), [DllCall](#), [NumGet](#), [NumPut](#)

## Examples

```

; SIMPLE STRUCTURE
pt:=Struct("x,y")
; SAME STRUCTURE CREATED FROM VARIABLE
_POINT:= "x,y"
pt:=Struct(_POINT)
; ARRAY OF POINTS
pt:=Struct("_POINT[10]")
; ARRAY OF POINTS OF UNKNOWN SIZE
pt:=Struct("*_POINT")

```

### ; TO CREATE ARRAY FROM POINTER

```
_POINT:="x,y", VarSetCapacity(pt1,8)
pt:=Struct("*_POINT") ; similar to "*_POINT[1]"
pt["",""]:=&pt1 ; assign pointer to first item
pt.x:=1 ; same as pt.1.x
MsgBox % pt.x
```

### ; Create a structure from structure

```
pt1:= new pt
pt1:= pt.Clone()
```

### ; More examples

```
pt:=Struct("x,y") ;POINT structure
pt.x:=100
MsgBox % pt.x
rc:=Struct("left,top,right,bottom") ; RECT
structure
Gui, Show, w100 h100, Test
Gui, +LastFound
DllCall("GetWindowRect", "PTR", WinExist(), "PTR", rc[
])
MsgBox % "left: " rc.left "`ntop: " rc.top
`nright: " rc.right "`nbottom: " rc.bottom
```

### ; Array Examples

#### ; Simple array structures.

#### ; Array is always accessed using integer

```
array:=Struct("Uint[10]")
array.5:=10
MsgBox % array.5
MyArray:="a,b"
array:=Struct("MyArray[10]")
array.1.a:=1
```

```
array.2.b:=2
MsgBox % array.1.a "`n" array.2.b
```

```
;
; Pointer Examples
; SIMPLE POINTER*
```

```
int:=Struct("*UInt"), VarSetCapacity(mem,100)
int["",""]:=&mem
int.1:=100 ; automatically resolves to pointer.
MsgBox % int.1 ; again pointer is resolved
```

```
automatically
```

```
; POINTER TO ARRAY OF POINTERS
```

```
VarSetCapacity(Arr,10*A_PtrSize,0)
Loop 10 ; create 10 arrays to hold 10 integers
 VarSetCapacity(Arr%A_Index%,10*sizeof("UInt"),0)
 , NumPut(&Arr%A_Index%,&arr,(A_Index-
1)*A_PtrSize,"PTR")
s:=Struct("**UInt")
s["",""]:=&Arr
s.1.1:=10
s.2.10:=20
MsgBox % s.1.1 "-" s.2.10
```

```
; ARRAY OF POINTERS
```

```
VarSetCapacity(Arr,10*A_PtrSize,0)
Loop 10 ; create 10 arrays to hold 10 integers
 VarSetCapacity(Arr%A_Index%,10*sizeof("UInt"),0)
 , NumPut(&Arr%A_Index%,&arr,(A_Index-
1)*A_PtrSize,"PTR")
s:=Struct("*UInt[10]",&Arr)
s.1.1:=30
s.2.10:=40
MsgBox % s.1.1 "-" s.2.10
```

```
; ARRAY OF POINTERS TO POINTERS
```

```
VarSetCapacity(Arr,10*A_PtrSize,0)
Loop 10 ; create 10 arrays to hold 10 arrays of 10
integers
```

```

{
 i:=A_Index
 VarSetCapacity(Arr%i%,10*A_PtrSize,0)
 , NumPut(&Arr%i%,&arr,(i-1)*4,"PTR")
 Loop 10

 VarSetCapacity(Arr%i%_A_Index%,10*sizeof("UInt"),
 0)
 , NumPut(&Arr%i%_A_Index%,&arr%i%,
 (A_Index-1)*4,"PTR")
}
s:=Struct("**UInt[10]",&Arr)
s.1.1.1:=50, s.2.3.10:=60
MsgBox % s.1.1.1 "-" s.2.3.10

```

```

; String Examples
; SIMPLE USER DEFINED STRUCTURE
user:="UInt Id, LPTSTR Name"
users := Struct("user[2]") ; array of structs
; here no automatic String allocation can be done
because users is evaluated dynamically so we use
own memory
Loop 2
 VarSetCapacity(Str%A_Index%,256)
 users.1.name[""]:=&Str1 ,users.2.name[""]:=&Str2
; above assignment can be assigned via object too
; users:=[{name:{"":&Str1}},{name:{"":&Str2}}]
users.1.Id := 1 ,users.2.Id := 2
users.1.Name := "Admin" ,users.2.Name := "User"
; same here, we could use an object to assign
values
; users:=[{id:1,name:"Admin"},{id:2,name:"User"}]
MsgBox % users.1.Id "`t" users.1.Name "`n"
users.2.Id "`t" users.2.Name
; Now to do the same with automatic String memory

```

we would need to use non dynamic structure  
; Therefore we define the structure completely and  
we can initialize it directly

```
users := Struct("UInt id1, LPTSTR Name1, UInt
id2, LPTSTR Name2",
{id1:1, name1:"Admin", id2:2, name2:"User"})
MsgBox % users.id1 "`t" users.name1 "`n" users.id2
"`t" users.name2
```

; CHAR ARRAY

```
String:=Struct("TCHAR char[26]")
Loop 26
 string["char"][A_Index]:=Chr(A_Index+64)
Loop 3
 MsgBox % String["char"][A_Index*2] ;show some
characters
MsgBox % StrGet(string[],26) ;get complete string
```

; RECT EXAMPLE

```
Gui,+LastFound
hwnd:=WinExist() ;get window handle
_RECT:="left,top,right,bottom"
RC:=Struct(_RECT) ;create structure
Gui,Add,Text,,Press Escape to continue
Gui,Show,w200 h100 ;show window
DllCall("GetWindowRect","PTR",hwnd,"PTR",rc[])
;get window position
rc.right := rc.right - rc.left ;Set rc.right to be
the width
rc.bottom := rc.bottom - rc.top ;Set rc.bottom to
be the height
While DllCall("GetCursorPos","PTR",rc[]) {
 DllCall("MoveWindow","PTR",hwnd,"int",rc.left,"in
t",rc.top,"int",rc.right,"int",rc.bottom,"Int",1)
 If GetKeyState("Escape","P")
 break
```

```
}
Gui, Destroy
```

```
;
; FINDFIRSTFILE
```

### EXAMPLE

```
_FILETIME := "dwLowDateTime,dwHighDateTime"
_SYSTEMTIME := "WORD wYear,WORD wMonth,WORD
wDayOfWeek,WORD wDay,WORD wHour,WORD wMinute,WORD
wSecond,WORD Milliseconds"
_WIN32_FIND_DATA := "dwFileAttributes,_FILETIME
ftCreationTime,_FILETIME
ftLastAccessTime,_FILETIME ftLastWriteTime,"
. .
"nFileSizeHigh,nFileSizeLow,dwReserved0,dwReserved
1,TCHAR cFileName[260],TCHAR
cAlternateFileName[14]"
file:=Struct("_WIN32_FIND_DATA[2]")
time:=Struct("_SYSTEMTIME")
DllCall("FindFirstFile","Str",A_ScriptFullPath,"Ui
nt",file.1[""])
DllCall("FindFirstFile","Str",A_AhkPath,"UInt",fil
e.2[""])
MsgBox % StrGet(file.1.cFileName[""])
MsgBox % "A_ScriptFullPath:`t"
StrGet(file.1.cFileName[""]) "`t"
StrGet(file.1.cAlternateFileName[""])
"`nA_AhkPath:`t" . StrGet(file.2.cFileName[""])
"`t" StrGet(file.2.cAlternateFileName[""])
handle:=DllCall("FindFirstFile","Str","C:*", "UInt
",file.2[""])
Loop {
 If
 !DllCall("FindNextFile","UInt",handle,"UInt",file.
2[""])
 break
 DllCall("FileTimeToSystemTime","UInt",file.2.ftLa
```

```

stWriteTime("", "Uint", time(""))
 ToolTip % StrGet(file.2.cFileName("")) "`n"
StrGet(file.2.cAlternateFileName("")) "`n"
file.2.nFileSizeHigh " - " file.2.nFileSizeLow
. "`n" time.wYear . "-" time.wMonth . "-"
time.wDay
. "`n" time.wDayOfWeek
. "`n" time.wHour . ":" time.wMinute . ":"
time.wSecond . ":" time.Milliseconds
 Sleep, 200
}
ToolTip
DllCall("CloseHandle", "Uint", handle)

```

```
;
```

**PROCESS32FIRST**

### EXAMPLE

```

MAX_PATH:=260
_PROCESSENTRY32:="
(
 DWORD dwSize;
 DWORD cntUsage;
 DWORD th32ProcessID;
 ULONG_PTR th32DefaultHeapID;
 DWORD th32ModuleID;
 DWORD cntThreads;
 DWORD th32ParentProcessID;
 LONG pcPriClassBase;
 DWORD dwFlags;
 TCHAR szExeFile[" MAX_PATH "];
)"
VarSetCapacity(string, 260)
pEntry:= Struct(_PROCESSENTRY32)
pEntry.dwSize := sizeof(_PROCESSENTRY32)
hSnapshot:=DllCall("CreateToolhelp32Snapshot", "UIn
t", TH32CS_SNAPALL:=0x0000001F, "PTR", 0)
DllCall("Process32First"

```

```

(A_IsUnicode?"w":""), "PTR", hSnapshot, "PTR", pEntry[
""])
While % (A_Index=1 || DllCall("Process32Next"
(A_IsUnicode?"w":""), "PTR", hSnapshot, "PTR", pEntry[
""])) {
 ToolTip % pEntry.cntUsage "`n"
pEntry.th32ProcessID
 . "`n" pEntry.th32DefaultHeapID "`n"
pEntry.th32ModuleID
 . "`n" pEntry.cntThreads "`n"
pEntry.th32ParentProcessID
 . "`n" pEntry.pcPriClassBase "`n"
pEntry.dwFlags "`n" StrGet(pEntry.szExeFile[""])
 Sleep, 200
}

```

```
;
```

**LISTPROCESSMODULES**

### EXAMPLE

```

MAX_PATH:=260
MAX_MODULE_NAME32:=255
_MODULEENTRY32:=
(
 DWORD dwSize;
 DWORD th32ModuleID;
 DWORD th32ProcessID;
 DWORD GblcntUsage;
 DWORD ProccntUsage;
 BYTE *modBaseAddr;
 DWORD modBaseSize;
 HMODULE hModule;
 TCHAR szModule[" MAX_MODULE_NAME32 + 1 "];
 TCHAR szExePath[" MAX_PATH "];
)"
ListProcessModules(DllCall("GetCurrentProcessId"))
ListProcessModules(dwPID){
 global _Struct

```

```

static
TH32CS_SNAPMODULE:=0x00000008,INVALID_HANDLE_VALUE
:=-1
hModuleSnap := Struct("HANDLE")
me32 := Struct("_MODULEENTRY32")
; Take a snapshot of all modules in the specified
process.
hModuleSnap :=
DllCall("CreateToolhelp32Snapshot","UInt",
TH32CS_SNAPMODULE,"PTR", dwPID)
if(hModuleSnap = INVALID_HANDLE_VALUE) {
 MsgBox % "CreateToolhelp32Snapshot (of
modules)"
 return FALSE
}
; Set the size of the structure before using it.
me32.dwSize := sizeof("_MODULEENTRY32")
; Retrieve information about the first module,
; and exit if unsuccessful
if(!DllCall("Module32First"
(A_IsUnicode?"W":""),"PTR", hModuleSnap,"PTR",
me32[""])) {
 MsgBox % "Module32First" ; Show cause of
failure
 DllCall("CloseHandle","PTR", hModuleSnap
) ; // Must clean up the snapshot object!
 return FALSE
}
; Now walk the module list of the process,
; and display information about each module
while(A_Index=1 || DllCall("Module32Next"
(A_IsUnicode?"W":""),"PTR",hModuleSnap,"PTR",
me32[""])) {
 ToolTip % "MODULE NAME`t=`t"
StrGet(me32.szModule[""])
 . "`nexecutable`t=`t"
StrGet(me32.szExePath[""])

```

```

 . "`nprocess ID`t=`t" me32.th32ProcessID
 . "`nref count (g)`t=`t"
me32.GlblCntUsage
 . "`nref count (p)`t=`t"
me32.ProcCntUsage
 . "`nbase address`t=`t"
me32.modBaseAddr[""]
 . "`nbase size`t=`t" me32.modBaseSize
 Sleep, 200
 }
; Do not forget to clean up the snapshot object.
DllCall("CloseHandle", "PTR", hModuleSnap)
return TRUE
}

```

**; Enumerate a structure.**

**; ENUMERATE SIMPLE STRUCTURE**

```

MyStruct:="a,b,c"
s:=Struct(MyStruct, {a:1,b:2,c:3})
for k, v in s
 MsgBox % k ": " v

```

**; ENUMERATE ARRAY OF STRUCTURES**

```

MyStruct:="a,b,c"
s:=Struct("MyStruct[3]", [{a:1,b:2,c:3},
{a:4,b:5,c:6},{a:7,b:8,c:9}])
for k, v in s
 for key,value in v
 MsgBox % key ": " value

```

**; ENUMERATE DYNAMIC STRUCTURE**

```

MyStruct:="a,b,c"
s:=Struct("Short size,LPTSTR name,MyStruct ms",
{size:sizeof(MyStruct),name:"MyStruct",ms:
{a:1,b:2,c:3}})
for k, v in s
 if !IsObject(v)

```

```
MsgBox % k ": " v
else
for key,value in v
 MsgBox % key ": " value
```

# sizeof

Built in function to calculate size of a structure or type like TCHAR or PTR or VOID..., see also [Struct](#) for usage and examples.

```
OutputVar := sizeof(Definition [, offset])
```

```
Function Example: sz := sizeof("Uint")
```

## Parameters

### OutputVar

The name of the variable in which to store the size of structure or data type.

### Definition

You can pass a [default data type](#) (e.g. "HANDLE") or structure definition (e.g. "int x,int y") as string or a structure object to retrieve size of a structure.

For a structure object you can also simply call `.Size()` method.

### offset

Offset of preceding structure. Used to calculate correct size and aligns when padding if necessary. This parameter is used internally and is not required to work with structures.

## Return Value

Size of structure. If offset was given it will return the new total size of structure (including offset).

## Related

[Struct](#), [DllCall](#)

## Examples

```
POINT:="UInt x,UInt y"
pt:=Struct(POINT)
MsgBox % sizeof(pt) " = " sizeof(POINT) " = "
pt.size()
```

# Built-in data types for Struct and sizeof functions.

AutoHotkey DllCall types are **bold**.

|               |              |           |          |   |
|---------------|--------------|-----------|----------|---|
| ACCESS_MASK   | ATOM         | BOOL      | BOOLEAN  | P |
| <b>DOUBLE</b> | DWORD        | DWORD_PTR | DWORD32  | I |
| HACCEL        | HALF_PTR     | HANDLE    | HBITMAP  | P |
| HCONVLIST     | HCURSOR      | HDC       | HDDEDATA | P |
| HENHMETAFILE  | HFILE        | HFONT     | HGDIOBJ  | P |
| HINSTANCE     | HKEY         | HKL       | HLOCAL   | P |
| HMONITOR      | HPALETTE     | HPEN      | HRESULT  | P |
| HWINSTA       | HWND         | INT       | INT8     | I |
| INT32         | <b>INT64</b> | LANGID    | LCID     | I |
|               |              |           |          |   |

|            |            |            |            |   |
|------------|------------|------------|------------|---|
| LONG_PTR   | LONG32     | LONG64     | LONGLONG   | I |
| LPCOLORREF | LPCSTR     | LPCTSTR    | LPCVOID    | I |
| LPINT      | LPLONG     | LPSTR      | LPTSTR     | I |
| LRESULT    | PBOOL      | PBOOLEAN   | PBYTE      | F |
| PCWSTR     | PDWORD     | PDWORD_PTR | PDWORD32   | F |
| PHALF_PTR  | PHANDLE    | PHKEY      | PINT       | F |
| PINT_PTR   | PINT32     | PINT64     | PLCID      | F |
| PLONG64    | PLONGLONG  | POINTER_32 | POINTER_64 | F |
| PSIZE_T    | PSSIZE_T   | PSTR       | PTBYTE     | F |
| PUCHAR     | PUHALF_PTR | PUINT      | PUINT_PTR  | F |
| PULONG_PTR | PULONG32   | PULONG64   | PULONGLONG | F |
| PWORD      | PWSTR      | SC_HANDLE  | SC_LOCK    | S |
|            |            |            |            |   |

|           |          |               |               |   |
|-----------|----------|---------------|---------------|---|
| SIZE_T    | SSIZE_T  | TBYTE         | TCHAR         | U |
| UINT_PTR  | UINT32   | <b>UINT64</b> | ULONG         | U |
| ULONGLONG | UNSIGNED | <b>UPTR</b>   | <b>USHORT</b> | U |
| WORD      | WPARAM   | __int64       |               |   |

## Related

[sizeof](#), [Struct](#), [DllCall](#), [NumGet](#), [NumPut](#)

# Control Functions

Functions to retrieve information about a control, or make a variety of changes to a control.

```
; General (all control types):
Boolean := ControlGetEnabled(...)
 ControlSetEnabled(TrueFalseToggle,
...)
Boolean := ControlGetVisible(...)
 ControlHide(...)
 ControlShow(...)
Integer := ControlGetHwnd(...)
Integer := ControlGetStyle(...)
 ControlSetStyle(Value, ...)
Integer := ControlGetExStyle(...)
 ControlSetExStyle(Value, ...)

; Edit:
Integer := ControlGetCurrentCol(...)
Integer := ControlGetCurrentLine(...)
Integer := ControlGetLineCount(...)
String := ControlGetLine(Index, ...)
String := ControlGetSelected(...)
 ControlEditPaste(String, ...)

; Checkbox and radio buttons:
Boolean := ControlGetChecked(...)
 ControlSetChecked(TrueFalseToggle,
...)

; ListBox and ComboBox:
String := ControlGetChoice(...)
 ControlChoose(Index, ...)
```

```

ControlChooseString(String, ...)
Index := ControlAddItem(String, ...)
ControlDeleteItem(Index, ...)
Index := ControlFindItem(String, ...)

; ListView, ListBox and ComboBox:
String := ControlGetList(Options, ...)

; ComboBox:
ControlShowDropDown(...)
ControlHideDropDown(...)

; Tab (SysTabControl32):
Integer := ControlGetTab(...)
ControlSetTab(N, ...)

```

Replace `...` with the [standard optional parameters](#):

`Control, WinTitle, WinText, ExcludeTitle, ExcludeText`

## All Controls

### ControlGetEnabled

Returns 1 (true) if *Control* is enabled, or 0 (false) if disabled.

```
ControlGetEnabled([Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

Control, WinTitle, etc.

See [Standard Parameters](#).

### ControlSetEnabled

Enables or disables a control.

```
ControlSetEnabled Value [, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

#### Value

One of the following case-insensitive strings or numbers:

**ON** or **1** (**true**) turns on the setting.

**OFF** or **0** (**false**) turns off the setting.

**TOGGLE** or **-1** sets it to the opposite of its current state.

Control, WinTitle, etc.

See [Standard Parameters](#).

## ControlGetVisible

Returns 1 (true) if *Control* is visible, or 0 (false) if hidden.

```
ControlGetVisible([Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

*Control*, *WinTitle*, etc.

See [Standard Parameters](#).

## ControlHide

Hides a control. If you additionally want to prevent a control's shortcut key (underlined letter) from working, disable the control via [ControlSetEnabled](#).

```
ControlHide [Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

*Control*, *WinTitle*, etc.

See [Standard Parameters](#).

## ControlShow

Shows a control if it was previously hidden.

```
ControlShow [Control, WinTitle, WinText,
```

`ExcludeTitle, ExcludeText`

[Control](#), [WinTitle](#), etc.

See [Standard Parameters](#).

## ControlGetHwnd

Retrieves the window handle (HWND) of the specified control.

`ControlGetHwnd`(`Control`, `WinTitle`, `WinText`,  
`ExcludeTitle`, `ExcludeText`)

[Control](#), [WinTitle](#), etc.

See [Standard Parameters](#).

For example: `editHwnd := ControlGetHwnd("Edit1",  
winTitle)`.

A control's HWND is often used with [PostMessage](#), [SendMessage](#), and [DllCall](#).  
On a related note, a control's HWND can also be retrieved via [MouseGetPos](#).  
Finally, a control's HWND can be used directly as an `ahk_id` [WinTitle](#) (this also works on hidden controls even when [DetectHiddenWindows](#) is Off).

## ControlGetStyle / ControlGetExStyle

Retrieves an integer representing the style or extended style of the control.

```
ControlGetStyle (Control, WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
ControlGetExStyle (Control, WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

Control, WinTitle, etc.

See [Standard Parameters](#).

See the [styles table](#) for a listing of some styles.

## ControlSetStyle / ControlSetExStyle

Changes the style or extended style of a control, respectively.

```
ControlSetStyle Value [, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
ControlSetExStyle Value [, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

### Value

Pass a positive integer to completely overwrite the window's style; that is, to set it to *Value*.

To easily add, remove or toggle styles, pass a numeric string prefixed with a plus sign (+), minus sign (-) or caret (^), respectively. The new style value is calculated as shown below (where CurrentStyle could be retrieved with [ControlGetStyle/ControlGetExStyle](#) or

WinGetStyle/WinGetExStyle):

| Operation | Prefix | Example String | Formula                                            |
|-----------|--------|----------------|----------------------------------------------------|
| Add       | +      | +0x80          | <code>NewStyle := CurrentStyle   Value</code>      |
| Remove    | -      | -0x80          | <code>NewStyle := CurrentStyle &amp; ~Value</code> |
| Toggle    | ^      | ^0x80          | <code>NewStyle := CurrentStyle ^ Value</code>      |

If *Value* is a negative integer, it is treated the same as the corresponding numeric string.

To use the + or ^ prefix literally in an expression, the prefix or value must be enclosed in quote marks. For example: `WinSetStyle("+0x80")` or `WinSetStyle("^" StylesToToggle)`. This is because the expression `+123` produces 123 (without a prefix) and `^123` is a syntax error.

**Control, WinTitle, etc.**

See [Standard Parameters](#).

`ErrorLevel` is set to 1 if the target window/control is not found or the style is not allowed to be applied.

Certain style changes require that the entire window be redrawn using `WinRedraw`. Also, the [styles table](#) lists some of the style numbers. For example:

```
ControlSetStyle("^0x800000", "Edit1",
"WinTitle") ; Set the WS_BORDER style to its
opposite state.
```

## CheckBox and Radio Buttons

### ControlGetChecked

Returns 1 (true) if the checkbox or radio button is checked or 0 (false) if not.

```
ControlGetChecked([Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

Control, WinTitle, etc.

See [Standard Parameters](#).

### ControlSetChecked

Turns on (checks) or turns off (unchecks) a radio button or checkbox.

```
ControlSetChecked Value [, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

#### Value

One of the following case-insensitive strings or numbers:

**ON** or **1** (**true**) turns on the setting.

**OFF** or **0** (**false**) turns off the setting.

**TOGGLE** or **-1** sets it to the opposite of its current state.

Control, WinTitle, etc.

See [Standard Parameters](#).

## Edit

### ControlGetCurrentCol

Returns the column number in an Edit control where the caret (text insertion point) resides.

```
ControlGetCurrentCol (Control, WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

Control, WinTitle, etc.

See [Standard Parameters](#).

The first column is 1. If there is text selected in the control, the return value is the column number where the selection begins.

### ControlGetCurrentLine

Returns the line number in an Edit control where the caret (insert point) resides.

```
ControlGetCurrentLine (Control, WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

Control, WinTitle, etc.

See [Standard Parameters](#).

The first line is 1. If there is text selected in the control, the return value is the

line number where the selection begins.

## ControlGetLineCount

Returns the number of lines in an Edit control.

```
ControlGetLineCount([Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

Control, WinTitle, etc.

See [Standard Parameters](#).

All Edit controls have at least 1 line, even if the control is empty.

## ControlGetLine

Returns the text of line *N* in an Edit control.

```
ControlGetLine(N [, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

**N**

The line number. Line 1 is the first line.

Control, WinTitle, etc.

See [Standard Parameters](#).

Depending on the nature of the control, the return value might end in a carriage

return `(r)` or a carriage return + linefeed `(r`n)`.

An [exception](#) is thrown if *N* negative or not a number.

```
For example: line1 := ControlGetLine(1, "Edit1",
"ahk_class Notepad")
```

## ControlGetSelected

Returns the selected text in an Edit control.

```
ControlGetSelected([Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

[Control](#), [WinTitle](#), etc.

See [Standard Parameters](#).

If no text is selected, an empty string is returned and `ErrorLevel` is set to 0 (i.e. no error). Certain types of controls, such as `RichEdit20A`, might not produce the correct text in some cases (e.g. `Metapad`).

## ControlEditPaste

Pastes *String* at the caret/insert position in an Edit control.

```
ControlEditPaste String [, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

## String

The string to paste.

## Control, WinTitle, etc.

See [Standard Parameters](#).

The effect is similar to pasting by pressing Ctrl+V, but this function does not affect the contents of the [clipboard](#) or require the control to have the keyboard focus.

## ListBox and ComboBox

### ControlGetChoice

Returns the name of the currently selected entry in a ListBox or ComboBox.

```
ControlGetChoice(Control, WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

Control, WinTitle, etc.

See [Standard Parameters](#).

To instead retrieve the position of the selected item, follow this example (use only one of the first two lines):

```
ChoicePos := SendMessage(0x188, 0, 0,
"ListBox1", WinTitle) ; 0x188 is LB_GETCURSEL
(for a ListBox).
ChoicePos := SendMessage(0x147, 0, 0,
"ComboBox1", WinTitle) ; 0x147 is CB_GETCURSEL
(for a DropDownList or ComboBox).
ChoicePos += 1 ; Convert from 0-based to 1-
based, i.e. so that the first item is known as
1, not 0.
; ChoicePos is now 0 if there is no item
selected.
```

### ControlChoose

Sets the selection in a ListBox or ComboBox to be the Nth entry. *N* should be 1 for the first entry, 2 for the second, etc.

```
ControlChoose N [, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

**N**

The index of the item, where 1 is the first entry, 2 is the second, etc.

To deselect all items, specify 0.

**Control, WinTitle, etc.**

See [Standard Parameters](#).

To select all items in a *multi-select* listbox, follow this example:

```
PostMessage, 0x185, 1, -1, ListBox1, WinTitle
; Select all listbox items. 0x185 is LB_SETSEL.
```

## **ControlChooseString**

Sets the selection in a ListBox or ComboBox to be the first entry whose leading part matches *String*.

```
ControlChooseString String [, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

**String**

The string to choose (see above). The search is not case sensitive. For example, if a ListBox/ComboBox contains the item "UNIX Text", specifying the word "unix" (lowercase) would be enough to select it.

Control, WinTitle, etc.

See [Standard Parameters](#).

Returns the index of the chosen item, where 1 is the first item, 2 is the second, etc. If an error occurred, the return value is blank and [ErrorLevel](#) is set to 1.

## ControlAddItem

Adds *String* as a new entry at the bottom of a ListBox or ComboBox.

```
N := ControlAddItem(String [, Control, WinTitle, WinText, ExcludeTitle, ExcludeText])
```

Control, WinTitle, etc.

See [Standard Parameters](#).

Returns the index of the new item, where 1 is the first item, 2 is the second, etc. If an error occurred, the return value is blank and [ErrorLevel](#) is set to 1.

## ControlDeleteItem

Deletes the Nth entry from a ListBox or ComboBox.

```
ControlDeleteItem N [, Control, WinTitle, WinText,
```

`ExcludeTitle, ExcludeText`

N

The index of the item, where 1 is the first entry, 2 is the second, etc.

[Control](#), [WinTitle](#), etc.

See [Standard Parameters](#).

## ControlFindItem

Returns the entry number of a ListBox or ComboBox that is a complete match for *String*.

`ControlFindItem` *String* [, [Control](#), [WinTitle](#), [WinText](#), [ExcludeTitle](#), [ExcludeText](#)]

[String](#)

The string to find. The search is case-insensitive. Unlike [ControlChooseString](#), the entry's entire text must match, not just the leading part.

[Control](#), [WinTitle](#), etc.

See [Standard Parameters](#).

The first entry in the control is 1, the second 2, and so on. If no match is found, the return value is blank and `ErrorLevel` is set to 1.

## ListView, ListBox and ComboBox

### ControlGetList

Retrieves a list of items from a ListView, ListBox, ComboBox, or DropDownList.

```
ControlGetList([Options, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

#### Options

Specifies what to retrieve if the control is a ListView (see below). For other control types, *Options* should be blank/empty.

#### Control, WinTitle, etc.

See [Standard Parameters](#).

### ListView

If the *Options* parameter is blank or omitted, all the text in the control is retrieved. Each row except the last will end with a linefeed character (`n). Within each row, each field (column) except the last will end with a tab character (`t).

Specify for *Options* zero or more of the following words, each separated from the next with a space or tab:

*Selected*: Retrieves only the selected (highlighted) rows rather than all rows. If

none, the return value is blank.

*Focused*: Retrieves only the focused row. If none, the return value is blank.

*Col4*: Retrieves only the fourth column (field) rather than all columns (replace 4 with a number of your choice).

*Count*: Retrieves a single number that is the total number of rows in the control.

*Count Selected*: Retrieves the number of selected (highlighted) rows.

*Count Focused*: Retrieves the row number (position) of the focused row (0 if none).

*Count Col*: Retrieves the number of columns in the control (or -1 if the count cannot be determined).

NOTE: Some applications store their ListView text privately, which prevents their text from being retrieved. In these cases, ErrorLevel will usually be set to 0 (indicating success) but all the retrieved fields will be empty.

Upon success, ErrorLevel is set to 0. Upon failure, it is set to 1 and the return value is blank. Failure occurs when: 1) the target window or control does not exist; 2) the target control is not of type SysListView32; 3) the process owning the ListView could not be opened, perhaps due to a lack of user permissions or because it is locked; 4) the *ColN option* specifies a nonexistent column.

To extract the individual rows and fields out of a ListView, use a [parsing loop](#) as in this example:

```
List := ControlGetList("Selected",
"SysListView321", WinTitle)
Loop, Parse, %List%, `n ; Rows are delimited
by linefeeds (`n).
```

```

{
 RowNumber := A_Index
 Loop, Parse, %A_LoopField%, %A_Tab%
 Fields (columns) in each row are delimited by
 tabs (A_Tab).
 MsgBox("Row #" RowNumber " Col #"
A_Index " is " A_LoopField ".")
}

```

On a related note, the columns in a ListView can be resized via [SendMessage](#) as shown in this example:

```

SendMessage(4126, 0, 80, "SysListView321",
WinTitle) ; 4126 is the message
LVM_SETCOLUMNWIDTH.
; In the above, 0 indicates the first column
(specify 1 for the second, 2 for the third,
etc.) Also, 80 is the new width.
; Replace 80 with -1 to autosize the column.
Replace it with -2 to do the same but also take
into account the header text width.

```

## ListBox and ComboBox (includes DropDownList)

All the text is retrieved from the control (that is, the ListView options above such as *Count* and *Selected* are not supported).

Each row except the last will be terminated by a linefeed character (^n). To access the items individually, use a [parsing loop](#) as in this example:

```

List := ControlGetList("", "ComboBox1",
WinTitle)
Loop, Parse, %List%, ^n

```



## ComboBox

### ControlShowDropDown

Drops a ComboBox so that its choices become visible.

```
ControlShowDropDown [Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

Control, WinTitle, etc.

See [Standard Parameters](#).

Example:

```
Send("#r") ; Display the Run dialog.
WinWaitActive("ahk_class #32770") ; Wait for
the dialog to appear.
ControlShowDropDown("ComboBox1") ; Show the
DropDown list. The fourth parameter is omitted
so that the last found window is used.
Sleep(2000)
ControlHideDropDown("ComboBox1") ; Hide the
DropDown list.
Sleep(1000)
Send("{Esc}") ; Hide the dialog window.
```

### ControlHideDropDown

Reverses the above.

**ControlHideDropDown** [Control, WinTitle, WinText,  
ExcludeTitle, ExcludeText]

Control, WinTitle, etc.

See [Standard Parameters](#).

## Tab

### ControlGetTab

Returns the position number of the selected tab in a SysTabControl32.

```
ControlGetTab (Control, WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

Control, WinTitle, etc.

See [Standard Parameters](#).

The first tab is 1, the second is 2, etc. If no tab is selected (rare), the return value is 0.

To instead discover how many tabs (pages) exist in a tab control, follow this example:

```
TabCount := SendMessage(0x1304,,,
"SysTabControl32", WinTitle) ; 0x1304 is
TCM_GETITEMCOUNT.
```

Example:

```
WhichTab := ControlGetTab("SysTabControl321",
"Some Window Title")
if ErrorLevel
 MsgBox("There was a problem.")
else
```

```
MsgBox("Tab #" WhichTab " is active.")
```

## ControlSetTab

Selects the Nth tab in a SysTabControl32.

```
ControlSetTab N [, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

Control, WinTitle, etc.

See [Standard Parameters](#).

## General

### Standard Parameters

All of the functions on this page utilize the following parameters to identify the target control and window:

#### Control

Can be either ClassNN (the classname and instance number of the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank, the target window's topmost control will be used.

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `"ahk_id " ControlHWND` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

#### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

#### WinText

If present, this parameter must be a substring from a single text element

of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise. Unless specified otherwise, each function also returns 1 (true) to indicate success or 0 (false) to indicate failure.

An [exception](#) is thrown if invalid parameters are detected.

## Remarks

To improve reliability, a delay is done automatically after each use of a Control function that changes a control (except for *ControlSetStyle* and *ControlSetExStyle*). That delay can be changed via [SetControlDelay](#) or by assigning a value to [A\\_ControlDelay](#). For details, see [SetControlDelay](#) remarks.

To discover the ClassNN or HWND of the control that the mouse is currently hovering over, use [MouseGetPos](#).

Window titles and text are case sensitive. Hidden windows are not detected

unless [DetectHiddenWindows](#) has been turned on.

## Related

[SetControlDelay](#), [WinSet](#) functions, [WinGet](#) functions, [GuiControl](#) object (for controls created by the script)

Other Control functions: [ControlGetFocus](#), [ControlFocus](#), [ControlGetPos](#), [ControlMove](#), [ControlGetText](#), [ControlSetText](#), [ControlClick](#), [ControlSend](#)

# ControlFocus

Sets input focus to a given control on a window.

```
ControlFocus [Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command example: ControlFocus "Edit1", "MyGui
ahk_class AutoHotkeyGUI"
```

```
Function example: ControlFocus("Edit1", "MyGui
ahk_class AutoHotkeyGUI")
```

## Parameters

### Control (optional)

Can be either ClassNN (the classname and instance number of the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank or omitted, the target window's topmost control will be used.

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle (optional)

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText (optional)

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle (optional)

Windows whose titles include this value will not be considered.

### ExcludeText (optional)

Windows whose text include this value will not be considered.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

## Remarks

To be effective, the control's window generally must not be minimized or hidden.

To improve reliability, a delay is done automatically after every use of this command. That delay can be changed via [SetControlDelay](#).

To discover the ClassNN or HWND of the control that the mouse is currently

hovering over, use [MouseGetPos](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[SetControlDelay](#), [ControlGetFocus](#), [Control](#) functions, [ControlMove](#), [ControlGetPos](#), [ControlClick](#), [ControlGetText](#), [ControlSetText](#), [ControlSend](#)

## Example

```
ControlFocus, OK, Some Window Title ; Set focus
to the OK button
```

# ControlGetFocus

Retrieves which control of the target window has input focus, if any.

```
OutPutVar := ControlGetFocus([WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Return Value

Returns the ClassNN of the control; that is, the control's classname followed by

its sequence number within its parent window, e.g. Button12.

If there was a problem or none of the target window's controls has focus, the return value is blank.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem determining the focus or 0 otherwise. An ErrorLevel of 1 typically indicates that the window does not exist.

## Remarks

The control retrieved by this command is the one that has keyboard focus, that is, the one that would receive keystrokes if the user were to type any.

The target window must be active to have a focused control, but even the active window may lack a focused control. If the target window was found but none of its controls have focus, ErrorLevel is set to 0 and the return value is blank.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[ControlFocus](#), [ControlMove](#), [ControlClick](#), [ControlGetText](#), [ControlSetText](#), [ControlSend](#)

## Example



# ControlGetPos

Retrieves the position and size of a control.

```
ControlGetPos [X, Y, Width, Height, Control,
WinTitle, WinText, ExcludeTitle, ExcludeText]
```

```
Command Example: ControlGetPos, x, y, w, h
"Edit1", "MyGui ahk_class AutoHotkeyGUI"
Function Example: ControlGetPos(x, y, w, h,
"Edit1", "MyGui ahk_class AutoHotkeyGUI")
```

## Parameters

### X, Y

The names of the variables in which to store the X and Y coordinates (in pixels) of *Control's* upper left corner. These coordinates are relative to the upper-left corner of the target window's [client area](#) and thus are the same as those used by [ControlMove](#).

If either X or Y is omitted, the corresponding values will not be stored.

### Width/Height

The names of the variables in which to store *Control's* width and height (in pixels). If omitted, the corresponding values will not be stored.

### Control

Can be either ClassNN (the classname and instance number of the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank, the target window's topmost control will be used.

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

If no matching window or control is found, the output variables will be made blank.

Unlike commands that change a control, `ControlGetPos` does not have an automatic delay (`SetControlDelay` does not affect it).

To discover the `ClassNN` or `HWND` of the control that the mouse is currently hovering over, use `MouseGetPos`. To retrieve a list of all controls in a window, use `WinGet`.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

`ControlMove`, `WinGetPos`, `Control` functions, `ControlGetText`, `ControlSetText`, `ControlClick`, `ControlFocus`, `ControlSend`

## Example

```
; This working example will continuously update
and display the
; name and position of the control currently under
the mouse cursor:
Loop
{
 Sleep, 100
 MouseGetPos, , , WhichWindow, WhichControl
```



# ControlGetText

Retrieves text from a control.

```
OutputVar := ControlGetText(Control, WinTitle,
WinText, ExcludeTitle, ExcludeText)
```

```
Function Example: Text :=
ControlGetText("Edit1", "MyGui ahk_class
AutoHotkeyGUI")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved text.

### Control

Can be either ClassNN (the classname and instance number of the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank or omitted, the target window's topmost control will be used.

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically

retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

## Remarks

Note: To retrieve text from a [ListView](#), [ListBox](#), or [ComboBox](#), use [ControlGetList](#) instead.

If the retrieved text appears to be truncated (incomplete), try using [VarSetCapacity\(OutputVar, 55\)](#) prior to [ControlGetText](#) [replace 55

with a size that is considerably longer than the truncated text]. This is necessary because some applications do not respond properly to the WM\_GETTEXTLENGTH message, which causes AutoHotkey to make the output variable too small to fit all the text.

This command might use a large amount RAM if the target control (e.g. an editor with a large document open) contains a large quantity of text. However, a variable's memory can be freed after use by assigning it to nothing, i.e.

```
OutputVar := "".
```

Text retrieved from most control types uses carriage return and linefeed (`\r\n`) rather than a solitary linefeed (`\n`) to mark the end of each line.

It is not necessary to do `SetTitleMatchMode Slow` because `ControlGetText` always retrieves the text using the slow method (since it works on a broader range of control types).

To retrieve a list of all controls in a window, use [WinGetControls](#) or [WinGetControlsHwnd](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[ControlSetText](#), [WinGetText](#), [Control functions](#), [ControlMove](#), [ControlFocus](#), [ControlClick](#), [ControlSend](#)

## Example

```
ControlGetText, OutputVar, Edit1, Untitled -
```

# ControlMove

Moves or resizes a control.

```
ControlMove [X, Y, width, Height, Control, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

```
Command Example: ControlMove 10, 10, 200, 20,
"Edit1", "MyGui ahk_class AutoHotkeyGUI"
```

```
Function Example: ControlMove(10, 10, 200, 20,
"Edit1", "MyGui ahk_class AutoHotkeyGUI")
```

## Parameters

### X, Y

The X and Y coordinates (in pixels) of the upper left corner of *Control's* new location. If either coordinate is blank, *Control's* position in that dimension will not be changed. The coordinates are relative to the upper-left corner of the target window's [client area](#); [ControlGetPos](#) can be used to determine them.

### Width, Height

The new width and height of *Control* (in pixels). If either parameter is blank or omitted, *Control's* size in that dimension will not be changed.

### Control

Can be either ClassNN (the classname and instance number of the

control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank, the target window's topmost control will be used.

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

To improve reliability, a delay is done automatically after every use of this command. That delay can be changed via `SetControlDelay`.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[ControlGetPos](#), [WinMove](#), [SetControlDelay](#), [Control functions](#), [ControlGetText](#), [ControlSetText](#), [ControlClick](#), [ControlFocus](#), [ControlSend](#)

## Example

### Function Syntax

```
SetTimer("ControlMoveTimer")
OutputVar := InputBox(, "My Input Box")
return

ControlMoveTimer:
If !WinExist("My Input Box")
 return
; Otherwise the above set the "last found" window
for us:
SetTimer("ControlMoveTimer", "off")
WinActivate()
ControlMove("OK", 10, "", 200) ; Move the OK
button to the left and increase its width.
```

```
return
```

## Command Syntax

```
SetTimer, ControlMoveTimer
InputBox, OutputVar,, My Input Box
return
```

```
ControlMoveTimer:
If !WinExist("My Input Box")
 return
```

```
; Otherwise the above set the "last found" window
for us:
```

```
SetTimer, ControlMoveTimer, off
WinActivate
ControlMove(10, , 200, , "OK") ; Move the OK
button to the left and increase its width.
return
```

# ControlSetText

Changes the text of a control.

```
ControlSetText NewText [, Control, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: ControlSetText "Hello
World!", "Edit1", "MyGui ahk_class
AutoHotkeyGUI"
```

```
Function Example: ControlSetText("Hello
World!", "Edit1", "MyGui ahk_class
AutoHotkeyGUI")
```

## Parameters

### NewText

The new text to set into the control. If blank or omitted, the control is made blank.

### Control

Can be either ClassNN (the classname and instance number of the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#). If this parameter is blank, the target window's topmost control will be used.

To operate upon a control's HWND (window handle), leave the *Control*

parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when *DetectHiddenWindows* is Off). The HWND of a control is typically retrieved via *ControlGetHwnd*, *MouseGetPos*, or *DllCall*.

### WinTitle

A window title or other criteria identifying the target window. See *WinTitle*.

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if *DetectHiddenText* is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

*ErrorLevel* is set to 1 if there was a problem or 0 otherwise.

## Remarks

Most control types use carriage return and linefeed (`\r\n`) rather than a solitary

linefeed (`\n`) to mark the end of each line. To translate a block of text containing `\n` characters, follow this example:

```
StrReplace, MyVar, MyVar, \n, \r\n, All
```

To improve reliability, a delay is done automatically after every use of this command. That delay can be changed via [SetControlDelay](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[SetControlDelay](#), [ControlGetFocus](#), [ControlGetText](#), [Control functions](#), [ControlMove](#), [ControlGetPos](#), [ControlClick](#), [ControlFocus](#), [ControlSend](#)

## Example

```
ControlSetText("New Text Here", "Edit1", "Untitled
-")
```

# MenuSelect

Invokes a menu item from the menu bar of the specified window.

```
MenuSelect WinTitle, WinText, Menu [, SubMenu1,
SubMenu2, SubMenu3, SubMenu4, SubMenu5, SubMenu6,
ExcludeTitle, ExcludeText]
```

```
Command Example: MenuSelect "A",, "File",
"Open"
Function Example: MenuSelect("A",, "File",
"Open")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### Menu

The name of the top-level menu, e.g. File, Edit, View. It can also be the position of the desired menu item by using 1& to represent the first menu,

2& the second, and so on.

### SubMenu1

The name of the menu item to select or its position (see above).

### SubMenu2

If *SubMenu1* itself contains a menu, this is the name of the menu item inside, or its position.

### SubMenu3

Same as above.

### SubMenu4

Same as above.

### SubMenu5

Same as above.

### SubMenu6

Same as above.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

For this command to work, the target window need not be active. However, some windows might need to be in a `non-minimized` state.

This command **will not work** with applications that use non-standard menu bars. Examples include Microsoft Outlook and Outlook Express, which use disguised toolbars for their menu bars. In these cases, consider using `ControlSend` or `PostMessage`, which should be able to interact with some of these non-standard menu bars.

The menu name parameters are not case sensitive (i.e. File->Save is the same as file->save) and the use of ampersand (&) to indicate the underlined letter in a menu item is not necessary (i.e. &File is the same as File).

The menu name parameters can also specify positions. This method exists to support menus that don't contain text (perhaps because they contain pictures of text rather than actual text). Position 1& is the first menu item (e.g. the File menu), position 2& is the second menu item (e.g. the Edit menu), and so on. Menu separator lines count as menu items for the purpose of determining the position of a menu item.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[ControlSend](#), [PostMessage](#)

## Example

```
; This will select File->Open in Notepad:
```

```
MenuSelect, Untitled - Notepad, , File, Open
```

```
; Same as above except it's done by position vs.
name:
```

```
MenuSelect, Untitled - Notepad, , 1&, 2&
```

# PostMessage / SendMessage

Sends a message to a window or control (SendMessage additionally waits for acknowledgement).

```
PostMessage Msg [, wParam, lParam, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]
```

```
SendMessage Msg [, wParam, lParam, Control, WinTitle, WinText, ExcludeTitle, ExcludeText, Timeout]
```

## Parameters

### Msg

The message number to send. See the [message list](#) to determine the number.

### wParam

The first component of the message. If blank or omitted, 0 will be sent.

### lParam

The second component of the message. If blank or omitted, 0 will be sent.

### Control

If this parameter is blank or omitted, the message will be sent directly to the target window rather than one of its controls. Otherwise, this parameter can be either ClassNN (the classname and instance number of

the control) or the control's text, both of which can be determined via Window Spy. When using text, the matching behavior is determined by [SetTitleMatchMode](#).

To operate upon a control's HWND (window handle), leave the *Control* parameter blank and specify `ahk_id %ControlHwnd%` for the *WinTitle* parameter (this also works on hidden controls even when [DetectHiddenWindows](#) is Off). The HWND of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#).

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

### Timeout

The maximum number of milliseconds to wait for the target window to

process the message. If omitted, it defaults to 5000 (milliseconds), which is also the default behaviour in older versions of AutoHotkey which did not support this parameter. If the message is not processed within this time, the command finishes and sets `ErrorLevel` to the word `ERROR`.

## ErrorLevel

`PostMessage`: `ErrorLevel` is set to 1 if there was a problem such as the target window or control not existing. Otherwise, it is set to 0.

`SendMessage`: `ErrorLevel` is set to the word `ERROR` if there was a problem or the command timed out. Otherwise, it is set to the numeric result of the message, which might sometimes be a "reply" depending on the nature of the message and its target window.

The range of possible values depends on the target window and the version of AutoHotkey that is running. When using a 32-bit version of AutoHotkey, or if the target window is 32-bit, the result is a 32-bit unsigned integer between 0 and 4294967295. When using the 64-bit version of AutoHotkey with a 64-bit window, the result is a 64-bit signed integer between -9223372036854775808 and 9223372036854775807.

If the result is intended to be a 32-bit signed integer (a value from -2147483648 to 2147483648), it can be truncated to 32-bit and converted to a signed value as follows:

```
MsgReply := ErrorLevel << 32 >> 32
```

This conversion may be necessary even on AutoHotkey 64-bit, because results from 32-bit windows are zero-extended. For example, a result of -1 from a 32-bit window is seen as 0xFFFFFFFF on any version of AutoHotkey, whereas a result of -1 from a 64-bit window is seen as 0xFFFFFFFF on AutoHotkey 32-bit and -1 on AutoHotkey 64-bit.

## Remarks

These commands should be used with caution because sending a message to the wrong window (or sending an invalid message) might cause unexpected behavior or even crash the target application. This is because most applications are not designed to expect certain types of messages from external sources.

PostMessage places the message in the message queue associated with the target window. It does not wait for acknowledgement or reply. By contrast, SendMessage waits for the target window to process the message, up until the timeout period expires.

The *wParam* and *lParam* parameters should be integers. If AutoHotkey or the target window is 32-bit, only the low 32 bits are used; that is, the value should be between -2147483648 and 4294967295 (0xFFFFFFFF). If AutoHotkey and the target window are both 64-bit, any integer value supported by AutoHotkey can be used. As with all integer values in AutoHotkey, a prefix of 0x indicates a hex value. For example, 0xFF is equivalent to 255.

A string may be sent via *wParam* or *lParam* by specifying the address of a variable. The following example uses the [address operator \(&\)](#) to do this:

```
SendMessage, 0xC, 0, &MyVar, ClassNN, WinTitle
; 0XC is WM_SETTEXT
```

A string put into MyVar by the receiver of the message is properly recognized without the need for extra steps. However, this works only if the parameter's first character is an ampersand (&); for example, `5+&MyVar` would not work but `&MyVar` or `&MyVar+5` would work.

A quoted/literal string may also be sent as in the following working example (the & operator should not be used in this case):

```
Run Notepad
WinWait Untitled - Notepad
SendMessage, 0xC, 0, "New Notepad Title" ; 0XC
is WM_SETTEXT
```

To send a message to all windows in the system, including those that are hidden or disabled, specify `ahk_id 0xFFFF` for *WinTitle* (0xFFFF is HWND\_BROADCAST). This technique should be used only for messages intended to be broadcast, such as the following example:

```
SendMessage, 0x1A,,,, ahk_id 0xFFFF ; 0x1A is
WM_SETTINGCHANGE
```

To have a script receive a message, use [OnMessage](#).

See the [Message Tutorial](#) for an introduction to using these commands.

Window titles and text are case sensitive. Hidden windows are not detected

unless [DetectHiddenWindows](#) has been turned on.

## Related

[Message List](#), [Message Tutorial](#), [OnMessage](#), [Automating Winamp](#), [DllCall](#), [ControlSend](#), [MenuSelect](#)

## Examples

```
#0:: ; Win+0 hotkey that turns off the monitor.
Sleep 1000 ; Give user a chance to release keys
(in case their release would wake up the monitor
again).
; Turn Monitor Off:
SendMessage, 0x112, 0xF170, 2,, Program Manager ;
0x112 is WM_SYSCOMMAND, 0xF170 is SC_MONITORPOWER.
; Note for the above: Use -1 in place of 2 to turn
the monitor on.
; Use 1 in place of 2 to activate the monitor's
low-power mode.
return

; Start the user's chosen screen saver:
SendMessage, 0x112, 0xF140, 0,, Program Manager ;
0x112 is WM_SYSCOMMAND, and 0xF140 is
SC_SCREENSAVE.

; Scroll up by one line (for a control that has a
vertical scroll bar):
ControlGetFocus, control, A
SendMessage, 0x115, 0, 0, %control%, A

; Scroll down by one line:
ControlGetFocus, control, A
```

```
SendMessage, 0x115, 1, 0, %control%, A
```

```
; Switch the active window's keyboard
layout/language to English:
```

```
PostMessage, 0x50, 0, 0x4090409,, A ; 0x50 is
WM_INPUTLANGCHANGEREQUEST.
```

```
; This example asks Winamp which track number is
currently active:
```

```
SetTitleMatchMode, 2
```

```
SendMessage, 1024, 0, 120, - Winamp
```

```
if ErrorLevel != "Error"
```

```
{
```

```
 ErrorLevel++ ; Winamp's count starts at "0",
so adjust by 1.
```

```
 MsgBox, Track #%ErrorLevel% is active or
playing.
```

```
}
```

```
; See Automating Winamp for more information.
```

```
; To find the process ID of an AHK script (an
alternative to "winGet PID"):
```

```
SetTitleMatchMode, 2
```

```
DetectHiddenWindows, on
```

```
SendMessage, 0x44, 0x405, 0, , SomeOtherScript.ahk
- AutoHotkey v
```

```
MsgBox %ErrorLevel% is the process id.
```

# SetControlDelay

Sets the delay that will occur after each control-modifying command.

**SetControlDelay** Delay

|                 |                 |                      |
|-----------------|-----------------|----------------------|
| <b>Command</b>  | <b>Example:</b> | SetControlDelay 100  |
| <b>Function</b> | <b>Example:</b> | SetControlDelay(100) |

## Parameters

### Delay

Time in milliseconds. Use -1 for no delay at all and 0 for the smallest possible delay. If unset, the default delay is 20.

## Remarks

A short delay (sleep) is done automatically after every Control function that changes a control. This is done to improve the reliability of scripts because a control sometimes needs a period of "rest" after being changed by one of these functions. The rest period allows it to update itself and respond to the next command that the script may attempt to send to it.

Specifically, SetControlDelay affects the following functions: [ControlAddItem](#), [ControlChoose](#), [ControlChooseString](#), [ControlClick](#), [ControlDeleteItem](#), [ControlEditPaste](#), [ControlFindItem](#), [ControlFocus](#), [ControlHide](#),

[ControlHideDropDown](#), [ControlMove](#), [ControlSetChecked](#), [ControlSetEnabled](#), [ControlSetTab](#), [ControlSetText](#), [ControlShow](#), [ControlShowDropDown](#).

[ControlSend](#) is not affected; it uses [SetKeyDelay](#).

Although a delay of -1 (no delay at all) is allowed, it is recommended that at least 0 be used, to increase confidence that the script will run correctly even when the CPU is under load.

A delay of 0 internally executes a `Sleep(0)`, which yields the remainder of the script's timeslice to any other process that may need it. If there is none, `Sleep(0)` will not sleep at all.

If the CPU is slow or under load, or if window animation is enabled, higher delay values may be needed.

The built-in variable `A_ControlDelay` contains the current setting and can also be assigned a new value instead of calling `SetControlDelay`.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[Control functions](#), [ControlMove](#), [ControlClick](#), [ControlFocus](#), [ControlSetText](#), [SetWinDelay](#), [SetKeyDelay](#), [SetMouseDelay](#)

## Example

```
SetControlDelay, 0
```

# GroupActivate

Activates the next window in a window group that was defined with [GroupAdd](#).

**GroupActivate** `GroupName` [`, R`]

|                          |                                       |
|--------------------------|---------------------------------------|
| <b>Command Example:</b>  | <code>GroupActivate "MyGroup"</code>  |
| <b>Function Example:</b> | <code>GroupActivate("MyGroup")</code> |

## Parameters

### GroupName

The name of the group to activate, as originally defined by [GroupAdd](#).

### R

This determines whether the oldest or the newest window is activated whenever no members of the group are currently active. If omitted, the oldest window is always activated. If it's the letter R, the newest window (the one most recently active) is activated, but only if no members of the group are active when the command is given. "R" is useful in cases where you temporarily switch to working on an unrelated task. When you return to the group via [GroupActivate](#), [GroupDeactivate](#), or [GroupClose](#), the window you were most recently working with is activated rather than the oldest window.

## ErrorLevel

`ErrorLevel` is set to 1 if no window was found to activate or 0 otherwise.

## Remarks

This command causes the first window that matches any of the group's window specifications to be activated. Using it a second time will activate the next window in the series and so on. Normally, it is assigned to a hotkey so that this window-traversal behavior is automated by pressing that key.

When a window is activated immediately after another window was activated, task bar buttons may start flashing on some systems (depending on OS and settings). To prevent this, use `#WinActivateForce`.

See `GroupAdd` for more details about window groups.

## Related

`GroupAdd`, `GroupDeactivate`, `GroupClose`, `#WinActivateForce`

## Example

```
GroupActivate, MyGroup, R
```

# GroupAdd

Adds a window specification to a window group, creating the group if necessary.

```
GroupAdd GroupName [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: GroupAdd "MyGroup",
"ahk_class AutoHotkeyGUI"
```

```
Function Example: GroupAdd("MyGroup", "ahk_class
AutoHotkeyGUI")
```

## Parameters

### GroupName

The name of the group to which to add this window specification. If the group doesn't exist, it will be created. Group names are not case sensitive.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON at the time that [GroupActivate](#), [GroupDeactivate](#), and [GroupClose](#) are used.

## ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Each use of this command adds a new rule to a group. In other words, a group consists of a set of criteria rather than a fixed list of windows. Later, when a group is used by a command such as [GroupActivate](#), each window on the desktop is checked against each of these criteria. If a window matches one of the criteria in the group, it is considered a match.

Although [SetTitleMatchMode](#) and [DetectHiddenWindows](#) do not directly affect the behavior of this command, they do affect the other group commands such as [GroupActivate](#) and [GroupClose](#). They also affect the use of `ahk_group` in any other command's [WinTitle](#).

A window group is typically used to bind together a collection of related windows, which is useful for tasks that involve many related windows, or an application that owns many subwindows. For example, if you frequently work with many instances of a graphics program or text editor, you can use [GroupActivate](#) on a hotkey to visit each instance of that program, one at a time, without having to use alt-tab or task bar buttons to locate them.

Since the entries in each group need to be added only once, this command is

typically used in the auto-execute section (top part of the script). Attempts to add duplicate entries to a group are ignored.

To include all windows in a group (except the special Program Manager window), use this example:

```
GroupAdd, AllWindows
```

All windowing commands can operate upon a window group by specifying `ahk_group MyGroupName` for the *WinTitle* parameter. The commands `WinMinimize`, `WinMaximize`, `WinRestore`, `WinHide`, `WinShow`, `WinClose`, and `WinKill` will act upon **all** the group's windows. To instead act upon only the topmost window, follow this example:

```
WinHide % "ahk_id " . WinExist("ahk_group
MyGroup")
```

By contrast, the other window commands such as `WinActivate` and `WinExist` will operate only upon the topmost window of the group.

## Related

`GroupActivate`, `GroupDeactivate`, `GroupClose`

## Examples

```
; In the autoexecute section at the top of the
script:
```

```
GroupAdd, MSIE, ahk_class IEFram ; Add only
Internet Explorer windows to this group.
return ; End of autoexecute section.
```

```
; Assign a hotkey to activate this group, which
traverses
; through all open MSIE windows, one at a time
(i.e. each
; press of the hotkey).
```

```
Numpad1::GroupActivate, MSIE, r
```

```
; Here's a more complex group for MS Outlook 2002.
; In the autoexecute section at the top of the
script:
```

```
SetTitleMatchMode, 2
```

```
GroupAdd, mail, Message - Microsoft Word ; This is
for mails currently being composed
```

```
GroupAdd, mail, - Message (; This is for already
opened items
```

```
; Need extra text to avoid activation of a phantom
window:
```

```
GroupAdd, mail, Advanced Find, Search for the
word(s)
```

```
GroupAdd, mail, , Recurrence:
```

```
GroupAdd, mail, Reminder
```

```
GroupAdd, mail, - Microsoft Outlook
```

```
return ; End of autoexecute section.
```

```
Numpad5::GroupActivate, mail ; Assign a hotkey to
visit each Outlook window, one at a time.
```

# GroupClose

Closes the active window if it was just activated by [GroupActivate](#) or [GroupDeactivate](#). It then activates the next window in the series. It can also close all windows in a group.

**GroupClose** GroupName [, A|R]

```
Command Example: GroupClose "MyGroup"
Function Example: GroupClose("MyGroup")
```

## Parameters

### GroupName

The name of the group as originally defined by [GroupAdd](#).

### A|R

If it's the letter A, all members of the group will be closed. This is the same effect as `winClose ahk_group GroupName`.

Otherwise: If the command closes the active window, it will then activate the next window in the series. This parameter determines whether the oldest or the newest window is activated. If omitted, the oldest window is always activated. If it's the letter R, the newest window (the one most recently active) is activated, but only if no members of the group are active when the command is given. "R" is useful in cases where you

temporarily switch to working on an unrelated task. When you return to the group via [GroupActivate](#), [GroupDeactivate](#), or [GroupClose](#), the window you were most recently working with is activated rather than the oldest window.

## Remarks

When the *A|R* parameter is not "A", the behavior of this command is determined by whether the previous action on *GroupName* was [GroupActivate](#) or [GroupDeactivate](#). If it was [GroupDeactivate](#), this command will close the active window only if it is **not** a member of the group (otherwise it will do nothing). If it was [GroupActivate](#) or nothing, this command will close the active window only if it **is** a member of the group (otherwise it will do nothing). This behavior allows [GroupClose](#) to be assigned to a hotkey as a companion to *GroupName*'s [GroupActivate](#) or [GroupDeactivate](#) hotkey.

See [GroupAdd](#) for more details about window groups.

## Related

[GroupAdd](#), [GroupActivate](#), [GroupDeactivate](#)

## Example

```
GroupClose, MyGroup, R
```

# GroupDeactivate

Similar to [GroupActivate](#) except activates the next window **not** in the group.

**GroupDeactivate** GroupName [ , R ]

```
Command Example: GroupDeactivate "MyGroup"
Function Example: GroupDeactivate("MyGroup")
```

## Parameters

### GroupName

The name of the target group, as originally defined by [GroupAdd](#).

### R

This determines whether the oldest or the newest non-member window is activated whenever a member of the group is currently active. If omitted, the oldest non-member window is always activated. If it's the letter R, the newest non-member window (the one most recently active) is activated, but only if a member of the group is active when the command is given. "R" is useful in cases where you temporarily switch to working on an unrelated task. When you return to the group via [GroupActivate](#), [GroupDeactivate](#), or [GroupClose](#), the window you were most recently working with is activated rather than the oldest window.

## Remarks

GroupDeactivate causes the first window that does **not** match any of the group's window specifications to be activated. Using GroupDeactivate a second time will activate the next window in the series and so on. Normally, GroupDeactivate is assigned to a hotkey so that this window-traversal behavior is automated by pressing that key.

This command is useful in cases where you have a collection of favorite windows that are almost always running. By adding these windows to a group, you can use GroupDeactivate to visit each window that isn't one of your favorites and decide whether to close it. This allows you to clean up your desktop much more quickly than doing it manually.

See [GroupAdd](#) for more details about window groups.

## Related

[GroupAdd](#), [GroupActivate](#), [GroupClose](#)

## Example

```
GroupDeactivate, MyFavoriteWindows ; Visit non-
favorite windows to clean up desktop.
```

# #WinActivateForce

Skips the gentle method of activating a window and goes straight to the forceful method.

## #WinActivateForce

Specifying this anywhere in a script will cause commands that activate a window -- such as [WinActivate](#), [WinActivateBottom](#), and [GroupActivate](#) -- to skip the "gentle" method of activating a window and go straight to the more forceful methods.

Although this directive will usually not change how quickly or reliably a window is activated, it might prevent task bar buttons from flashing when different windows are activated quickly one after the other.

## Related

[WinActivate](#), [WinActivateBottom](#), [GroupActivate](#)

## Example

```
#WinActivateForce
```

# DetectHiddenText

Determines whether invisible text in a window is "seen" for the purpose of finding the window. This affects commands such as WinExist and WinActivate.

**DetectHiddenText** On|Off

|                          |                        |
|--------------------------|------------------------|
| <b>Command Example:</b>  | DetectHiddenText "On"  |
| <b>Function Example:</b> | DetectHiddenText("On") |

## Parameters

On|Off

**On** or 1 (*true*): Hidden text is detected. This is the default.

**Off** or 0 (*false*): Hidden text is not detected.

## Remarks

"Hidden text" is a term that refers to those controls of a window that are not visible. Their text is thus considered "hidden". Turning off DetectHiddenText can be useful in cases where you want to detect the difference between the different panes of a multi-pane window or multi-tabbed dialog. Use Window Spy to determine which text of the currently-active window is hidden. All commands that accept a *WinText* parameter are affected by this setting, including WinActivate, WinActive, WinWait, and WinExist.

The built-in variable **A\_DetectHiddenText** contains the current setting (On or Off).

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[DetectHiddenWindows](#)

## Example

```
DetectHiddenText, off
```

# DetectHiddenWindows

Determines whether invisible windows are "seen" by the script.

**DetectHiddenWindows** On|Off

|                 |                 |                           |
|-----------------|-----------------|---------------------------|
| <b>Command</b>  | <b>Example:</b> | DetectHiddenWindows "On"  |
| <b>Function</b> | <b>Example:</b> | DetectHiddenWindows("On") |

## Parameters

On|Off

**On** or 1 (*true*): Hidden windows are detected.

**Off** or 0 (*false*): This is the default. Hidden windows are not detected, except by the [WinShow](#) command.

## Remarks

Turning on DetectHiddenWindows can make scripting harder in some cases since some hidden system windows might accidentally match the title or text of another window you're trying to work with. So most scripts should leave this setting turned off. However, turning it on may be useful if you wish to work with hidden windows directly without first using [WinShow](#) to unhide them.

All windowing commands except [WinShow](#) are affected by this setting, including [WinActivate](#), [WinActive](#), [WinWait](#) and [WinExist](#). By contrast,

`WinShow` will always unhide a hidden window even if hidden windows are not being detected.

Turning on `DetectHiddenWindows` is not necessary when accessing a control or child window via the `ahk_id` method or as the `last-found-window`. It is also not necessary when accessing GUI windows via `+LastFound` option.

The built-in variable `A_DetectHiddenWindows` contains the current setting (On or Off).

Every newly launched `thread` (such as a `hotkey`, `custom menu item`, or `timed subroutine`) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[DetectHiddenText](#)

## Example

```
DetectHiddenWindows, on
```

# SetTitleMatchMode

Sets the matching behavior of the *WinTitle* parameter in commands such as [WinWait](#).

```
SetTitleMatchMode MatchMode
SetTitleMatchMode "Fast|Slow"
```

## Parameters

### MatchMode

One of the following digits or the word *RegEx*:

- 1: A window's title must start with the specified *WinTitle* to be a match.
- 2: A window's title can contain *WinTitle* anywhere inside it to be a match.
- 3: A window's title must exactly match *WinTitle* to be a match.

**RegEx:** Changes *WinTitle*, *WinText*, *ExcludeTitle*, and *ExcludeText* to accept [regular expressions](#). Do not enclose such expressions in quotes when using them with commands. For example: `winActivate Untitled.\*Notepad`.

Note:

- RegEx also applies to `ahk_class` and `ahk_exe`; for example, `ahk_class IEFrame` searches for any window whose class name contains *IEFrame* anywhere (this is because by default,

regular expressions find a match *anywhere* in the target string).

- For *WinTitle*, each component is separate. For example, in `i)^untitled ahk_class i)^notepad$ ahk_pid %mypid%`, `i)^untitled` and `i)^notepad$` are separate regex patterns and `%mypid%` is always compared numerically (it is not a regex pattern).
- For *WinText*, each text element (i.e. each control's text) is matched against the RegEx separately. Therefore, it is not possible to have a match span more than one text element.

The modes above also affect *ExcludeTitle* in the same way as *WinTitle*. For example, mode 3 requires that a window's title exactly match *ExcludeTitle* for that window to be excluded.

### Fast|Slow

One of the following words to specify how the *WinText* and *ExcludeText* parameters should be matched:

**Fast:** This is the default behavior. Performance may be substantially better than *Slow*, but certain types of controls are not detected. For instance, text is typically detected within Static and Button controls, but not Edit controls, unless they are owned by the script.

**Slow:** Can be much slower, but works with all controls which respond to the `WM_GETTEXT` message.

### Remarks

This command affects the behavior of all windowing commands, e.g. `WinExist` and `WinActivate`. `WinGetText` is affected in the same way as other commands, but it always uses the *Slow* method to retrieve text.

If unspecified, `TitleMatchMode` defaults to 2 and *fast*.

If a `window group` is used, the current title match mode applies to each individual rule in the group.

Generally, the *slow* mode should be used only if the target window cannot be uniquely identified by its title and *fast*-mode text. This is because the slow mode can be extremely slow if there are any application windows that are busy or "not responding".

Window Spy has an option for *Slow TitleMatchMode* so that its easy to determine whether the *Slow* mode is needed.

If you wish to change both attributes, run the command twice as in this example:

```
SetTitleMatchMode, 2
SetTitleMatchMode, slow
```

The built-in variables `A_TitleMatchMode` and `A_TitleMatchModeSpeed` contain the current settings.

Regardless of the current `TitleMatchMode`, `WinTitle`, `WinText`, `ExcludeTitle` and `ExcludeText` are case sensitive. The only exception is the `case-insensitive` option of the RegEx mode; for example: `|i)untitled - notepad|`.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[SetWinDelay](#), [WinExist](#), [WinActivate](#), [RegExMatch](#)

## Example

```
SetTitleMatchMode 1
; OR:
SetTitleMatchMode RegEx

SetTitleMatchMode Slow ; Slow/Fast can be set
independently of all the other modes.
```

# SetWinDelay

Sets the delay that will occur after each windowing command, such as [WinActivate](#).

**SetWinDelay** Delay

```
Command Example: SetWinDelay 200
Function Example: SetWinDelay(200)
```

## Parameters

### Delay

Time in milliseconds. Use -1 for no delay at all and 0 for the smallest possible delay. If unset, the default delay is 100.

## Remarks

A short delay (sleep) is done automatically after every windowing command except [WinActive](#) and [WinExist](#). This is done to improve the reliability of scripts because a window sometimes needs a period of "rest" after being created, activated, minimized, etc. so that it has a chance to update itself and respond to the next command that the script may attempt to send to it.

Although a delay of -1 (no delay at all) is allowed, it is recommended that at least 0 be used, to increase confidence that the script will run correctly even

when the CPU is under load.

A delay of 0 internally executes a `Sleep(0)`, which yields the remainder of the script's timeslice to any other process that may need it. If there is none, `Sleep(0)` will not sleep at all.

If the CPU is slow or under load, or if window animation is enabled, higher delay values may be needed.

The built-in variable `A_WinDelay` contains the current setting.

Every newly launched [thread](#) (such as a [hotkey](#), [custom menu item](#), or [timed subroutine](#)) starts off fresh with the default setting for this command. That default may be changed by using this command in the auto-execute section (top part of the script).

## Related

[SetControlDelay](#), [SetKeyDelay](#), [SetMouseDelay](#), [SendMode](#)

## Example

```
SetWinDelay, 10
```

# StatusBarGetText

Retrieves the text from a standard status bar control.

```
OutputVar := StatusBarGetText(Part#, WinTitle,
WinText, ExcludeTitle, ExcludeText)
```

```
Function Example: text := StatusBarGetText(1,
"A")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved text.

### Part#

Which part number of the bar to retrieve. Default 1, which is usually the part that contains the text of interest.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility).

Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

[ErrorLevel](#) is set to 1 if there is a problem or 0 otherwise. If there was a problem, *OutputVar* is also made blank.

## Remarks

This command 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, in which case the text cannot be retrieved.

Rather than using this command in a loop, it is usually more efficient to use [StatusBarWait](#), which contains optimizations that avoid the overhead of repeated calls to `StatusBarItem.GetText`.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

StatusBarWait, WinGetTitle, WinGetText, ControlGetText

## Example

```
StatusBarGetText, RetrievedText, 1, Search Results
if InStr(RetrievedText, "found"), MsgBox("Search
results have been found.")
```

# StatusBarWait

Waits until a window's status bar contains the specified string.

```
StatusBarWait [BarText, Seconds, Part#, WinTitle,
WinText, Interval, ExcludeTitle, ExcludeText]
```

## Parameters

### BarText

The text or partial text for the which the command will wait to appear. Default is blank (empty), which means to wait for the status bar to become blank. The text is case sensitive and the matching behavior is determined by [SetTitleMatchMode](#), similar to *WinTitle* below.

To instead wait for the bar's text to *change*, either use [StatusBarGetText](#) in a loop, or use the RegEx example at the bottom of this page.

### Seconds

The number of seconds (can contain a decimal point) to wait before timing out, in which case [ErrorLevel](#) will be set to 1. Default is blank, which means wait indefinitely. Specifying 0 is the same as specifying 0.5.

### Part#

Which part number of the bar to retrieve. Default 1, which is usually the part that contains the text of interest.

## WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

## WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

## Interval

How often the status bar should be checked while the command is waiting (in milliseconds). Default is 50.

## ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

[ErrorLevel](#) is set to 1 if the command times out before a match could be found in the status bar. It is set to 2 if the status bar could not be accessed. It is set to 0 if a match is found.

## Remarks

StatusBarWait attempts to read the first *standard* status bar on a window (class `msctls_statusbar32`). Some programs use their own status bars or special versions of the MS common control. Such bars are not supported.

Rather than using `StatusBarGetText` in a loop, it is usually more efficient to use `StatusBarWait` because it contains optimizations that avoid the overhead that repeated calls to `StatusBarGetText` would incur.

`StatusBarWait` determines its target window before it begins waiting for a match. If that target window is closed, the command will stop waiting even if there is another window matching the specified `WinTitle` and `WinText`.

While the command is in a waiting state, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[StatusBarGetText](#), [WinGetTitle](#), [WinGetText](#), [ControlGetText](#)

## Example

```
; The following example enters a new search
pattern into an existing Explorer/Search window.
if WinExist("Search Results") ; Sets the Last
Found window to simplify the below.
{
 WinActivate
```

```

Send, {tab 2}!o*.txt{enter} ; In the Search
window, enter the pattern to search for.
Sleep, 400 ; Give the status bar time to
change to "Searching".
StatusBarWait, found, 30
if ErrorLevel
 MsgBox, The command timed out or there was
a problem.
else
 MsgBox, The search successfully completed.
}

```

; The following example waits for the status bar of the active window to change.

```

SetTitleMatchMode RegEx
if WinExist("A") ; Set the last-found window to
be the active window (for use below).
{
 StatusBarGetText, OrigText
 StatusBarWait, ^(?!\^\Q%OrigText%\E$) ; This
regular expression waits for any change to the
text.
}

```

# WinActivate

Activates the specified window (makes it foremost).

```
WinActivate [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

```
Command Example: WinActivate "ahk_class
AutoHotkeyGUI"
```

```
Function Example: WinActivate("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

If the window is minimized, it is automatically restored prior to being activated. In v1.1.20 and later, the window is restored even if it is already active.

Six attempts will be made to activate the target window over the course of 60ms. Thus, it is usually unnecessary to follow `WinActivate` with `WinWaitActive` or `!WinActive(...)`.

If a matching window is already active, that window will be kept active rather than activating any other matching window beneath it. In general, if more than one window matches, the topmost (most recently used) will be activated. You can activate the bottommost (least recently used) via `WinActivateBottom`.

If the active window is hidden and `DetectHiddenWindows` is turned off, it is never considered a match. Instead, a visible matching window is activated if one exists.

When a window is activated immediately after the activation of some other window, task bar buttons might start flashing on some systems (depending on OS and settings). To prevent this, use `#WinActivateForce`.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

**Known issue:** If the script is running on a computer or server being accessed via

remote desktop, WinActivate may hang if the remote desktop client is minimized. One workaround is to use commands which don't require window activation, such as [ControlSend](#) and [ControlClick](#). Another possible workaround is to apply the following registry setting on the local/client computer:

```
; Change HKCU to HKLM to affect all users on
this system.
RegWrite REG_DWORD,
HKCU\Software\Microsoft\Terminal Server Client
 , RemoteDesktop_SuppressWhenMinimized, 2
```

## Related

[WinActivateBottom](#), [#WinActivateForce](#), [SetTitleMatchMode](#),  
[DetectHiddenWindows](#), [Last Found Window](#), [WinExist](#), [WinActive](#),  
[WinWaitActive](#), [WinWait](#), [WinWaitClose](#), [WinClose](#), [GroupActivate](#)

## Example

```
If WinExist("Untitled - Notepad")
 WinActivate ; use the window found above
else
 WinActivate, Calculator
```

# WinActivateBottom

Same as [WinActivate](#) except that it activates the bottommost (least recently active) matching window rather than the topmost.

```
WinActivateBottom [WinTitle, WinText, ExcludeTitle, ExcludeText]
```

```
Command Example: WinActivateBottom "ahk_class AutoHotkeyGUI"
Function Example: WinActivateBottom("ahk_class AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

If there is only one matching window, `WinActivateBottom` behaves identically to `WinActivate`.

`Window groups` are more advanced than this command, so consider using them for more features and flexibility.

If the window is minimized, it is automatically restored prior to being activated.

Six attempts will be made to activate the target window over the course of 60ms. Thus, it is usually unnecessary to follow it with the `WinWaitActive` command.

Unlike `WinActivate`, the `Last Found Window` cannot be used because it might not be the bottommost window. Therefore, at least one of the parameters must be non-blank.

When a window is activated immediately after another window was activated, task bar buttons may start flashing on some systems (depending on OS and settings). To prevent this, use `#WinActivateForce`.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

WinActivate, #WinActivateForce, SetTitleMatchMode, DetectHiddenWindows, WinExist, WinActive, WinWaitActive, WinWait, WinWaitClose, GroupActivate

## Example

```
; This hotkey allows you to visit all open browser windows in order from oldest to newest:
```

```
#i::
SetTitleMatchMode, 2
WinActivateBottom, - Microsoft Internet Explorer
return
```

# WinActive

Checks if the specified window exists and is currently active (foremost). If it is, *WinActive()* returns its Unique ID (HWND).

```
OutputVar := WinActive (WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Function Example: id := WinActive("A")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved [UniqueID](#).

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Return Value

Returns the [Unique ID \(HWND\)](#) of the active window if it matches the specified criteria, or 0 if it does not.

Since all non-zero numbers are seen as "true", the statement *if WinActive("WinTitle")* is true whenever *WinTitle* is active.

## Remarks

If all parameters are omitted, the [Last Found Window](#) will be used.

If the active window is a qualified match, the [Last Found Window](#) will be updated to be the active window.

An easy way to retrieve the unique ID of the active window is with

```
ActiveHwnd := WinExist("A").
```

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinExist](#), [SetTitleMatchMode](#), [DetectHiddenWindows](#), [Last Found Window](#), [WinActivate](#), [WinWaitActive](#), [WinWait](#), [WinWaitClose](#), [#IfWinActive/Exist](#)

## Example

```
If WinActive("Untitled - Notepad")
{
 WinMaximize ; Maximizes the Notepad window
 found by WinActive above.
 Send, Some text.{Enter}
 return
}

if WinActive("ahk_class Notepad") or
WinActive("ahk_class" . ClassName) ; "ahk_class"
need not have a space after it.
 WinClose ; Uses the last found window.
```

# WinClose

Closes the specified window.

```
WinClose WinTitle, WinText, SecondsToWait,
ExcludeTitle, ExcludeText
```

```
Command Example: WinClose "A"
Function Example: WinClose("A")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### SecondsToWait

If omitted or blank, the command will not wait at all. If 0, it will wait 500ms. Otherwise, it will wait the indicated number of seconds (can contain a decimal point or be an [expression](#)) for the window to close. If the window does not close within that period, the script will continue.

ErrorLevel is **not** set by this command, so use [WinExist](#) or [WinWaitClose](#) if you need to determine for certain that a window is closed. While the command is in a waiting state, new [threads](#) can be launched via [hotkey](#), [custom menu item](#), or [timer](#).

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

This command sends a close message to a window. The result depends on the window (it may ask to save data, etc.)

If a matching window is active, that window will be closed in preference to any other matching window beneath it. In general, if more than one window matches, the topmost (most recently used) will be closed.

This command operates only upon the topmost matching window except when *WinTitle* is [ahk\\_group GroupName](#), in which case all windows in the group are affected.

WinClose sends a WM\_CLOSE message to the target window, which is a somewhat forceful method of closing it. An alternate method of closing is to send the following message. It might produce different behavior because it is similar in effect to pressing Alt-F4 or clicking the window's close button in its

title bar:

```
PostMessage, 0x112, 0xF060,,, WinTitle, WinText
; 0x112 = WM_SYSCOMMAND, 0xF060 = SC_CLOSE
```

If a window does not close via WinClose, you can force it to close with [WinKill](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinKill](#), [WinWaitClose](#), [ProcessClose](#), [WinActivate](#), [SetTitleMatchMode](#),  
[DetectHiddenWindows](#), [Last Found Window](#), [WinExist](#), [WinActive](#),  
[WinWaitActive](#), [WinWait](#), [GroupActivate](#)

## Example

```
If WinExist("Untitled - Notepad")
 WinClose ; use the window found above
else
 WinClose, Calculator
```

# WinExist

Checks if a matching window exists. *WinExist()* returns the [Unique ID \(HWND\)](#) of the first matching window.

```
OutputVar := WinExist(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Function Example: id := WinExist("A")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved [UniqueID](#).

### UniqueID

[Unique ID \(HWND\)](#) of the first window matching the given criteria.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included [Window Spy](#) utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

## ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Return Value

Returns the [Unique ID \(HWND\)](#) of the first matching window (or 0 if none).

Since all non-zero numbers are seen as "true", the statement *if WinExist("WinTitle")* is true whenever *WinTitle* exists.

## Remarks

If all parameters are omitted, the [Last Found Window](#) will be checked to see if it still exists.

If a qualified window exists, the [Last Found Window](#) will be updated to be that window.

To discover the HWND of a control (for use with [Post/SendMessage](#) or [DllCall](#)), use [ControlGetHwnd](#) or [MouseGetPos](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

WinActive, SetTitleMatchMode, DetectHiddenWindows, Last Found Window, ProcessExist, WinActivate, WinWaitActive, WinWait, WinWaitClose, #IfWinActive/Exist

## Examples

**; Example 1**

```
if WinExist("Untitled - Notepad")
{
 WinActivate ; Automatically uses the window
found above.
 WinMaximize ; same
 Send, Some text.{Enter}
 return
}
```

**; Example 2**

```
if !WinExist("Calculator")
 return
else
{
 WinActivate ; The above "IfWinNotExist" also
set the "last found" window for us.
 WinMove, 40, 40 ; Move it to a new position.
 return
}
```

**; Example 3**

```
if WinExist("ahk_class Notepad") or
WinExist("ahk_class" . ClassName)
 WinActivate ; Uses the last found window.

MsgBox % "The active window's ID is " .
WinExist("A")
```

```
; Example 4: Equivalent to IfWinNotExist,
Calculator
```

```
If !WinExist("Calculator")
 return
```

# WinGet

Functions for retrieving information about a window, or a list of windows. These functions all have the same syntax, as below; just replace *WinGetX* with the appropriate function name.

```
OutputVar := WinGetX([WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

## Parameters

### OutputVar

The name of the variable in which to store the result of the command. When calling the command as a function, *OutputVar* represents its return value.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Functions

**WinGetID:** Equivalent to [WinExist](#). Retrieves the unique ID number (HWND/handle) of a window. If there is no matching window, *OutputVar* is made blank. `WinGetID("A")` is a fast way to get the ID of the active window. To discover the HWND of a control (for use with [Post/SendMessage](#) or [DllCall](#)), use [ControlGetHwnd](#) or [MouseGetPos](#).

**WinGetIDLast:** Same as above except it retrieves the ID of the last/bottommost window if there is more than one match. If there is only one match, it performs identically to *ID*. This concept is similar to that used by [WinActivateBottom](#).

**WinGetPID:** Retrieves the [Process ID \(PID\)](#) of a window.

**WinGetProcessName:** Retrieves the name of the process (e.g. notepad.exe) that owns a window. If there are no matching windows, *OutputVar* is made blank.

**WinGetProcessPath:** Similar to *ProcessName*, but retrieves the full path and name of the process instead of just the name.

**WinGetCount:** Retrieves the number of existing windows that match the specified *WinTitle*, *WinText*, *ExcludeTitle*, and *ExcludeText* (0 if none). To count all windows on the system, omit all four title/text parameters. Hidden windows

are included only if [DetectHiddenWindows](#) has been turned on.

**WinGetList:** Retrieves an [array](#) of the unique ID numbers of all existing windows that match the specified *WinTitle*, *WinText*, *ExcludeTitle*, and *ExcludeText*. To retrieve all windows on the entire system, omit all four title/text parameters.

**WinGetMinMax:** Retrieves the minimized/maximized state for a window. *OutputVar* is made blank if no matching window exists; otherwise, it is set to one of the following numbers:

-1: The window is minimized ([WinRestore](#) can unminimize it).

1: The window is maximized ([WinRestore](#) can unmaximize it).

0: The window is neither minimized nor maximized.

**WinGetControls:** Retrieves an [array](#) of control names for all controls in a window. If no matching window exists, *OutputVar* is made blank. If there are no controls in the window, the result is an empty array. Otherwise, each control name consists of its class name followed immediately by its sequence number (ClassNN), as shown by Window Spy.

Controls are sorted according to their Z-order, which is usually the same order as TAB key navigation if the window supports tabbing.

The control currently under the mouse cursor can be retrieved via [MouseGetPos](#).

**WinGetControlsHwnd:** Same as above except it retrieves the [window handle \(HWND\)](#) of each control rather than its ClassNN.

**WinGetTransparent:** Retrieves the degree of transparency of a window (see [WinSetTransparent](#) for how to set transparency). *OutputVar* is made blank if: 1) the OS is older than Windows XP; 2) there are no matching windows; 3) the window has no transparency level; or 4) other conditions (caused by OS behavior) such as the window having been minimized, restored, and/or resized since it was made transparent. Otherwise, a number between 0 and 255 is stored, where 0 indicates an invisible window and 255 indicates an opaque window. For example:

```
MouseGetPos,,, MouseWin
WinGetTransparent, Transparent, ahk_id
%MouseWin% ; Transparency of window under the
mouse cursor.
```

**WinGetTransColor:** Retrieves the color that is marked transparent in a window (see [WinSetTransColor](#) for how to set the TransColor). *OutputVar* is made blank if: 1) the OS is older than Windows XP; 2) there are no matching windows; 3) the window has no transparent color; or 4) other conditions (caused by OS behavior) such as the window having been minimized, restored, and/or resized since it was made transparent. Otherwise, a six-digit hexadecimal RGB color is stored, e.g. 0x00CC99. For example:

```
MouseGetPos,,, MouseWin
WinGetTransColor, TransColor, ahk_id %MouseWin%
; TransColor of the window under the mouse
cursor.
```

**WinGetStyle** or **WinGetExStyle:** Retrieves an integer representing the style or

extended style (respectively) of a window. If there are no matching windows, *OutputVar* is made blank. The following example determines whether a window has the `WS_DISABLED` style:

```
WinGetStyle, Style, My Window Title
if (Style & 0x80000000) ; 0x80000000 is
WS_DISABLED.
... the window is disabled, so perform
appropriate action.
```

The next example determines whether a window has the `WS_EX_TOPMOST` style (always-on-top):

```
WinGetExStyle, ExStyle, My Window Title
if (ExStyle & 0x8) ; 0x8 is WS_EX_TOPMOST.
... the window is always-on-top, so perform
appropriate action.
```

See the [styles table](#) for a partial listing of styles.

## Remarks

A window's ID number is valid only during its lifetime. In other words, if an application restarts, all of its windows will get new ID numbers.

ID numbers retrieved by these commands are numeric (the prefix "ahk\_id" is not included).

The ID of the window under the mouse cursor can be retrieved with [MouseGetPos](#).

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[WinGetClass](#), [Process](#), [WinGetTitle](#), [MouseGetPos](#), [Control](#) functions, [ControlFocus](#), [GroupAdd](#)

## Examples

```
; Example #1: Maximize the active window and report its unique ID:
```

```
WinGetID, active_id, A
WinMaximize, ahk_id %active_id%
MsgBox, The active window's ID is "%active_id%".
```

```
; Example #2: This will visit all windows on the entire system and display info about each of them:
```

```
WinGetList, id,,, Program Manager
Loop % id.Length
{
 this_id := id[A_Index]
 WinActivate, ahk_id %this_id%
 WinGetClass, this_class, ahk_id %this_id%
 WinGetTitle, this_title, ahk_id %this_id%
 Result := MsgBox("Visiting All
Windows`n%a_index% of %id.Length%`nahk_id
%this_id%`nahk_class
%this_class%`n%this_title%`n`nContinue?",, 4)
 if Result = "No", break
}
```

```
; Example #3: Display each of a window's control
```

**names:**

```
for n, ctrl in WinGetControls("A")
{
 Result := MsgBox("Control #%%n% is \"%ctrl%".
Continue?",, 4)
 if Result = "No"
 break
}
```

**; Example #4: Display in real time the active window's control list:**

```
SetTimer, WatchActiveWindow, 200
```

```
WatchActiveWindow() {
 ControlList := ""
 WinGetControls, Controls, A
 if !Controls {
 ToolTip, No visible window is active.
 return
 }
 if !Controls.Length() {
 ToolTip, The active window has no
controls.
 return
 }
 Loop % Controls.Length()
 ControlList .= Controls[A_Index] . "`n"
 ToolTip, %ControlList%
}
```

# WinGetClass

Retrieves the specified window's class name.

```
OutputVar := WinGetClass(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Function Example: class := WinGetClass("A")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved class name.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Only the class name is retrieved (the prefix "ahk\_class" is not included in *OutputVar*).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinGet](#), [WinGetTitle](#)

## Example

```
WinGetClass, class, A
MsgBox, The active window's class is "%class%".
```

# WinGetPos

Retrieves the position and size of the specified window.

```
WinGetPos [X, Y, Width, Height, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinGetPos x, y, w, h,
"ahk_class AutoHotkeyGUI"
Function Example: WinGetPos(x, y, w, h,
"ahk_class AutoHotkeyGUI")
```

## Parameters

### X, Y

The names of the variables in which to store the X and Y coordinates of the target window's upper left corner. If omitted, the corresponding values will not be stored.

### Width/Height

The names of the variables in which to store the width and height of the target window. If omitted, the corresponding values will not be stored.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

## WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

## ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

If no matching window is found, the output variables will be made blank.

If the *WinTitle* "Program Manager" is used, the command will retrieve the size of the desktop, which is usually the same as the current screen resolution.

A minimized window will still have a position and size. The values returned in this case may vary depending on OS and configuration.

To discover the name of the window and control that the mouse is currently hovering over, use [MouseGetPos](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

WinMove, ControlGetPos, WinGetTitle, WinGetText, ControlGetText

## Example

```
WinGetPos, X, Y, Width, Height, Calculator
MsgBox, Calculator is at %X%\,%Y%

WinGetPos, X, Y, , , A ; "A" to get the active
window's pos.
MsgBox, The active window is at %X%\,%Y%

if WinExist("Untitled - Notepad")
{
 WinGetPos, Xpos, Ypos ; Uses the window found
above.
 MsgBox, Notepad is at %Xpos%\,%Ypos%
}
```

# WinGetText

Retrieves the text from the specified window.

```
OutputVar := WinGetText([WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

```
Function Example: text := WinGetText("A")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved text.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if there was a problem or 0 otherwise.

## Remarks

The text retrieved is generally the same as what Window Spy shows for that window. However, if `DetectHiddenText` has been turned off, hidden text is omitted from *OutputVar*.

Each text element ends with a carriage return and linefeed (CR+LF), which can be represented in the script as ``r`n`. To extract individual lines or substrings, use functions such as `InStr` and `SubStr`. A [parsing loop](#) can also be used to examine each line or word one by one.

If the retrieved text appears to be truncated (incomplete), try using `VarSetCapacity(OutputVar, 55)` prior to `WinGetText` [replace 55 with a size that is considerably longer than the truncated text]. This is necessary because some applications do not respond properly to the `WM_GETTEXTLENGTH` message, which causes AutoHotkey to make the output variable too small to fit all the text.

This command might use a large amount of RAM if the target window (e.g. an editor with a large document open) contains a large quantity of text. To avoid

this, it might be possible to retrieve only portions of the window's text by using `ControlGetText` instead. In any case, a variable's memory can be freed later by assigning it to nothing, i.e. `OutputVar := ""`.

To retrieve a list of all controls in a window, follow this example:

```
WinGetControls, OutputVar, WinTitle
```

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[ControlGetText](#), [WinGetTitle](#), [WinGetPos](#)

## Example

```
Run, Calc.exe
WinWait, Calculator
WinGetText, text ; The window found above will be
used.
MsgBox, The text is:`n%text%
```

# WinGetTitle

Retrieves the title of the specified window.

```
OutputVar := WinGetTitle(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Function Example: title := WinGetTitle("A")
```

## Parameters

### OutputVar

The name of the variable in which to store the retrieved title.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

To discover the name of the window that the mouse is currently hovering over, use [MouseGetPos](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinSetTitle](#), [WinGetClass](#), [WinGet](#), [WinGetText](#), [ControlGetText](#), [WinGetPos](#)

## Example

```
WinGetTitle, Title, A
MsgBox, The active window is "%Title%".
```

# WinHide

Hides the specified window.

```
WinHide [WinTitle, WinText, ExcludeTitle, ExcludeText]
```

```
Command Example: WinHide "A"
Function Example: WinHide("A")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Use [WinShow](#) to unhide a hidden window ([DetectHiddenWindows](#) can be either On or Off to do this).

This command operates only upon the topmost matching window except when [WinTitle](#) is [ahk\\_group GroupName](#), in which case all windows in the group are affected.

The Explorer taskbar may be hidden/shown as follows:

```
WinHide ahk_class Shell_TrayWnd
WinShow ahk_class Shell_TrayWnd
```

## Related

[WinShow](#), [SetTitleMatchMode](#), [DetectHiddenWindows](#), [Last Found Window](#)

## Example

```
Run, notepad.exe
WinWait, Untitled - Notepad
Sleep, 500
WinHide ; use the window found above
Sleep, 1000
WinShow
```

# WinKill

Forces the specified window to close.

```
WinKill [WinTitle, WinText, SecondsToWait,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinKill "A"
Function Example: WinKill("A")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### SecondsToWait

If omitted or blank, the command will not wait at all. If 0, it will wait 500ms. Otherwise, it will wait the indicated number of seconds (can contain a decimal point or be an [expression](#)) for the window to close. If the window does not close within that period, the script will continue.

ErrorLevel is **not** set by this command, so use [WinExist](#) or [WinWaitClose](#) if you need to determine for certain that a window is closed.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

This command first makes a brief attempt to close the window normally. If that fails, it will attempt to force the window closed by terminating its process.

If a matching window is active, that window will be closed in preference to any other matching window beneath it. In general, if more than one window matches, the topmost (most recently used) will be closed.

This command operates only upon the topmost matching window except when *WinTitle* is `ahk_group GroupName`, in which case all windows in the group are affected.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinClose](#), [WinWaitClose](#), [ProcessClose](#), [WinActivate](#), [SetTitleMatchMode](#),

DetectHiddenWindows, Last Found Window, WinExist, WinActive,  
WinWaitActive, WinWait, GroupActivate

## Example

```
If WinExist("Untitled - Notepad")
 WinKill ; use the window found above
else
 WinKill, Calculator
```

# WinMaximize

Enlarges the specified window to its maximum size.

```
WinMaximize [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

```
Command Example: WinMaximize "ahk_class
AutoHotkeyGUI"
```

```
Function Example: WinMaximize("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Use [WinRestore](#) to unmaximize a window and [WinMinimize](#) to minimize it.

If a particular type of window does not respond correctly to [WinMaximize](#), try using the following instead:

```
PostMessage, 0x112, 0xF030,,, WinTitle, WinText
; 0x112 = WM_SYSCOMMAND, 0xF030 = SC_MAXIMIZE
```

This command operates only upon the topmost matching window except when *WinTitle* is `ahk_group GroupName`, in which case all windows in the group are affected.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinRestore](#), [WinMinimize](#)

## Example

```
Run, notepad.exe
WinWait, Untitled - Notepad
WinMaximize ; use the window found above

^Up::WinMaximize, A ; Assign a hotkey to maximize
```

the active window.

# WinMinimize

Collapses the specified window into a button on the task bar.

```
WinMinimize [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

```
Command Example: WinMinimize "ahk_class
AutoHotkeyGUI"
```

```
Function Example: WinMinimize("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Use [WinRestore](#) or [WinMaximize](#) to unminimize a window.

If a particular type of window does not respond correctly to [WinMinimize](#), try using the following instead:

```
PostMessage, 0x112, 0xF020,,, WinTitle, WinText
; 0x112 = WM_SYSCOMMAND, 0xF020 = SC_MINIMIZE
```

This command operates only upon the topmost matching window except when *WinTitle* is `ahk_group GroupName`, in which case all windows in the group are affected.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinRestore](#), [WinMaximize](#), [WinMinimizeAll](#)

## Example

```
Run, notepad.exe
WinWait, Untitled - Notepad
WinMinimize ; use the window found above

^Down::WinMinimize, A ; Assign a hotkey to
```

minimize the active window.

# WinMinimizeAll / WinMinimizeAllUndo

Minimizes or unminimizes all windows.

## **WinMinimizeAll**

WinMinimizeAllUndo

|                 |                 |                  |
|-----------------|-----------------|------------------|
| <b>Command</b>  | <b>Example:</b> | WinMinimizeAll   |
| <b>Function</b> | <b>Example:</b> | WinMinimizeAll() |

On most systems, this is equivalent to Explorer's Win-M and Win-D hotkeys.

## **Related**

[WinMinimize](#), [GroupAdd](#)

## **Example**

```
WinMinimizeAll
WinMinimizeAllUndo
```

# WinMove

Changes the position and/or size of the specified window.

```
WinMove X, Y [, Width, Height, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinMove 100, 200
Function Example: WinMove(100, 200)
```

## Parameters

### X, Y

The X and Y coordinates (in pixels) of the upper left corner of the target window's new location. The upper-left pixel of the screen is at 0, 0.

If these are the only parameters given with the command, the [Last Found Window](#) will be used as the target window.

Otherwise, X and/or Y can be omitted, in which case the current position is used.

### Width, Height

The new width and height of the window (in pixels). If either is omitted

or blank, the size in that dimension will not be changed.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

If *Width* and *Height* are small (or negative), most windows with a title bar will generally go no smaller than 112 x 27 pixels (however, some types of windows may have a different minimum size). If *Width* and *Height* are large, most windows will go no larger than approximately 12 pixels beyond the dimensions of the desktop.

Negative values are allowed for the x and y coordinates to support multi-monitor systems and to allow a window to be moved entirely off screen.

Although WinMove cannot move minimized windows, it can move hidden windows if [DetectHiddenWindows](#) is on.

The speed of WinMove is affected by [SetWinDelay](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[ControlMove](#), [WinGetPos](#), [WinHide](#), [WinMinimize](#), [WinMaximize](#)

## Example

```
Run("calc.exe")
WinWaitActive("Calculator")
WinMove(0, 0) ; Move the window found by WinWait
to the upper-left corner of the screen.

Gui := GuiCreate("ToolWindow -System Menu Disabled",
"The clipboard contains:")
Gui.Add("Text",,, Clipboard)
Gui.Show("w400 h300")
WinMove(Gui.Title,, 0, 0) ; Move the splash window
to the top left corner.
Msgbox("Press OK to dismiss the GUI window")
Gui.Destroy()

; The following function centers the specified
window on the screen:
CenterWindow(WinTitle)
{
 WinGetPos(,, Width, Height, WinTitle)
```



# WinRestore

Unminimizes or unmaximizes the specified window if it is minimized or maximized.

```
WinRestore [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

```
Command Example: WinRestore "ahk_class
AutoHotkeyGUI"
Function Example: WinRestore("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

If a particular type of window does not respond correctly to WinRestore, try using the following instead:

```
PostMessage, 0x112, 0xF120,,, WinTitle, WinText
; 0x112 = WM_SYSCOMMAND, 0xF120 = SC_RESTORE
```

This command operates only upon the topmost matching window except when *WinTitle* is `ahk_group GroupName`, in which case all windows in the group are affected.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[WinMinimize](#), [WinMaximize](#)

## Example

```
WinRestore, Untitled - Notepad
```

# WinSet

Functions for making a variety of changes to a window, such as "always on top" and transparency.

## WinSetAlwaysOnTop

Makes a window stay on top of all other windows (except other always-on-top windows).

```
WinSetAlwaysOnTop [Value, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

### Value

One of the following case-insensitive text values or numbers:

`ON` or `1` (`true`) turns on the setting.

`OFF` or `0` (`false`) turns off the setting.

`TOGGLE` or `-1` sets it to the opposite of its current state.

If omitted, it defaults to `TOGGLE`.

`WinGetExStyle` can be used to determine whether a window is always-on-top.

**Related:** [Window Parameters](#), [Return Value](#)

## WinMoveBottom

Sends a window to the bottom of stack; that is, beneath all other windows. The effect is similar to pressing Alt-Escape.

```
WinMoveBottom [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

**Related:** [Window Parameters](#), [Return Value](#)

## WinMoveTop

Brings a window to the top of the stack without explicitly [activating](#) it.

```
WinMoveTop [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

The system default settings will probably cause it to activate in most cases. In addition, this command may have no effect due to the operating system's protection against applications that try to steal focus from the user (it may depend on factors such as what type of window is currently active and what the user is currently doing). One possible work-around is to make the window briefly [AlwaysOnTop](#), then turn off [AlwaysOnTop](#).

**Related:** [Window Parameters](#), [Return Value](#)

## WinSetEnabled

Enables or disables a window.

```
WinSetEnabled Value [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

### Value

One of the following case-insensitive text values or numbers:

`ON` or `1` (`true`) turns on the setting.

`OFF` or `0` (`false`) turns off the setting.

`TOGGLE` or `-1` sets it to the opposite of its current state.

If omitted, it defaults to `TOGGLE`.

When a window is disabled, the user cannot move it or interact with its controls. In addition, disabled windows are omitted from the alt-tab list.

`WinGetStyle` can be used to determine whether a window is disabled.

**Related:** [Window Parameters](#), [Return Value](#)

## WinRedraw

Attempts to update the appearance/contents of a window by informing the OS that the window's rectangle needs to be redrawn.

```
WinRedraw [WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

If this method does not work for a particular window, try [WinMove](#). If that does

not work, try the following:

```
WinHide WinTitle
WinShow WinTitle
```

**Related:** [Window Parameters](#), [Return Value](#)

## WinSetStyle / WinSetExStyle

Changes the style or extended style of a window.

```
WinSetStyle Value [, WinTitle, WinText, ExcludeTitle,
ExcludeText]
```

```
WinSetExStyle Value [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

### Value

Pass a positive integer to completely overwrite the window's style; that is, to set it to *Value*.

To easily add, remove or toggle styles, pass a numeric string prefixed with a plus sign (+), minus sign (-) or caret (^), respectively. The new style value is calculated as shown below (where *CurrentStyle* could be retrieved with [WinGetStyle/WinGetExStyle](#)):

| Operation | Prefix | Example String | Formula                             |
|-----------|--------|----------------|-------------------------------------|
| Add       | +      | +0x80          | NewStyle :=<br>CurrentStyle   Value |

|        |   |       |                                                    |
|--------|---|-------|----------------------------------------------------|
| Remove | - | -0x80 | <code>NewStyle := CurrentStyle &amp; ~Value</code> |
| Toggle | ^ | ^0x80 | <code>NewStyle := CurrentStyle ^ Value</code>      |

If *Value* is a negative integer, it is treated the same as the corresponding numeric string.

To use the + or ^ prefix literally in an expression, the prefix or value must be enclosed in quote marks. For example: `WinSetStyle("+0x80")` or `WinSetStyle("^" StylesToToggle)`. This is because the expression `+123` produces 123 (without a prefix) and `^123` is a syntax error.

After applying certain style changes to a visible window, it might be necessary to redraw the window using `WinRedraw`. Finally, the [styles table](#) lists some of the common style numbers. Examples:

```
WinSetStyle, -0xC00000, A ; Remove the active
window's title bar (WS_CAPTION).
WinSetExStyle, ^0x80, WinTitle ; Toggle the
WS_EX_TOOLWINDOW attribute, which removes/adds
the window from the alt-tab list.
```

**Related:** [Window Parameters](#), [Return Value](#), [WinGetStyle/WinGetExStyle](#)

## WinSetRegion

Changes the shape of a window to be the specified rectangle, ellipse, or polygon.

```
WinSetRegion [Options, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

If the *Options* parameter is blank, the window is restored to its original/default display area. Otherwise, one or more of the following options can be specified, each separated from the others with space(s):

**Wn**: Width of rectangle or ellipse. For example: `w200`.

**Hn**: Height of rectangle or ellipse. For example: `h300`.

**X-Y**: Each of these is a pair of X/Y coordinates. For example, `200-0` would use 200 for the X coordinate and 0 for the Y.

**E**: Makes the region an ellipse rather than a rectangle. This option is valid only when **W** and **H** are present.

**R[w-h]**: Makes the region a rectangle with rounded corners. For example, `R30-30` would use a 30x30 ellipse for each corner. If **w-h** is omitted, 30-30 is used. **R** is valid only when **W** and **H** are present.

**Rectangle or ellipse**: If the **W** and **H** options are present, the new display area will be a rectangle whose upper left corner is specified by the first (and only) pair of **X-Y** coordinates. However, if the **E** option is also present, the new display area will be an ellipse rather than a rectangle. For example:

```
WinSetRegion, 50-0 W200 H250 E, WinTitle.
```

**Polygon**: When the **W** and **H** options are absent, the new display area will be a polygon determined by multiple pairs of **X-Y** coordinates (each pair of coordinates is a point inside the window relative to its upper left corner). For

example, if three pairs of coordinates are specified, the new display area will be a triangle in most cases. The order of the coordinate pairs with respect to each other is sometimes important. In addition, the word **Wind** maybe be present in *Options* to use the winding method instead of the alternating method to determine the polygon's region.

`ErrorLevel` and the function's `return value` are set based on whether the function succeeds or fails. Failure occurs when: 1) the target window does not exist; 2) one or more *Options* are invalid; 3) more than 2000 pairs of coordinates were specified; or 4) the specified region is invalid or could not be applied to the target window.

See the bottom of this page for examples of how to use this command.

**Related:** [Window Parameters](#), [Return Value](#)

## WinSetTransparent

Makes a window semi-transparent.

```
WinSetTransparent [N, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

**N**

To enable transparency, specify a number between 0 and 255 indicating the degree of transparency: 0 makes the window invisible while 255 makes it opaque.

To turn off transparency completely, specify the word `OFF` (case-insensitive).

Turning off transparency is different than specifying 255 because it may improve performance and reduce usage of system resources (but probably only on Windows XP, or when desktop composition is disabled).

### Known Limitations for Transparent and `TransColor`:

- Setting "Transparent" to 255 prior to turning it off might avoid window redrawing problems such as a black background. If the window still fails to be redrawn correctly, see [Redraw](#) for a possible workaround.
- To change a window's existing `TransColor`, it may be necessary to turn off transparency before making the change.

**Tip:** To make the task bar transparent, use `WinSetTransparent, 150, ahk_class Shell_TrayWnd`. Similarly, to make the Start Menu transparent, follow this example:

```
DetectHiddenWindows, on
WinSetTransparent, 150, ahk_class BaseBar ; To
make the Start Menu's submenus transparent,
also include the script below.
```

To make all or selected menus on the entire system transparent, keep a script such as the following always running. Note that although such a script cannot make its own menus transparent, it can make those of other scripts transparent:

```
SetTimer, WatchForMenu, 5
return ; End of auto-execute section.

WatchForMenu:
DetectHiddenWindows, on ; Might allow
detection of menu sooner.
If WinExist("ahk_class #32768")
 WinSetTransparent, 150 ; Uses the window
found by the above line.
return
```

**Related:** [Window Parameters](#), [Return Value](#), [WinGetTransparent](#)

## WinSetTransColor

```
WinSetTransColor Color [N] [, Options, WinTitle,
WinText, ExcludeTitle, ExcludeText]
```

Makes all pixels of the chosen color invisible inside the target window, which allows the contents of the window behind it to show through. If the user clicks on an invisible pixel, the click will "fall through" to the window behind it. Specify for *Color* a color name or RGB value (see the [color chart](#) for guidance, or use [PixelGetColor](#) in its RGB mode). To additionally make the visible part of the window partially transparent, append a space (not a comma) followed by the transparency level (0-255). For example: `WinSetTransColor, EEAA99 150, WinTitle`.

TransColor is often used to create on-screen displays and other visual effects. There is an example of an on-screen display [at the bottom of the Gui page](#).

The word OFF may be specified to completely turn off transparency for a window. Both of the following are identical in function:

```
WinSetTransparent, Off, WinTitle
WinSetTransparentColor, Off, WinTitle
```

Known Limitations: See the list [above](#).

**Related:** [Window Parameters](#), [Return Value](#), [WinGetTransparentColor](#)

## Window Parameters

All of the functions on this page utilize the following parameters to identify the target window:

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Return Value

All of the functions on this page return 1 ([true](#)) on success and 0 ([false](#)) on failure. The return value is only accessible when using the function syntax.

Conversely, [ErrorLevel](#) is set to 0 ([false](#)) on success and 1 ([true](#)) on failure. [ErrorLevel](#) is always set.

Failure occurs if the target window is not found or the change could not be applied.

## Remarks

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinGet](#), [WinHide](#), [WinSetTitle](#), [WinMove](#), [WinActivate](#), [Control functions](#)

## Examples

```
WinSetTransparent, 200, Untitled - Notepad ; Make
the window a little bit transparent.
WinSetTransColor, White, Untitled - Notepad ; Make
all white pixels invisible.
WinSetAlwaysOnTop, toggle, Calculator ; Toggle the
always-on-top status of Calculator.
```

```
; Longer Example:
; Here are some hotkeys that demonstrate the
effects of "Transparent" and
; "TransColor". NOTE: If you press one of the
hotkeys while the mouse cursor
; is hovering over a pixel that is invisible as a
result of TransColor, the window
; visible beneath that pixel will be acted upon
instead!
```

```
#t:: ; Press Win+T to make the color under the
mouse cursor invisible.
```

```
MouseGetPos, MouseX, MouseY, MouseWin
PixelGetColor, MouseRGB, %MouseX%, %MouseY%, RGB
```

```
; In seems necessary to turn off any existing
transparency first:
```

```
WinSetTransColor, Off, ahk_id %MouseWin%
WinSetTransColor, %MouseRGB% 220, ahk_id
%MouseWin%
return
```

```
#o:: ; Press Win+O to turn off transparency for
the window under the mouse.
```

```
MouseGetPos,,, MouseWin
WinSetTransColor, Off, ahk_id %MouseWin%
return
```

```
#g:: ; Press Win+G to show the current settings
of the window under the mouse.
```

```
MouseGetPos,,, MouseWin
WinGetTransparent, Transparent, ahk_id %MouseWin%
WinGetTransColor, TransColor, ahk_id %MouseWin%
ToolTip
Translucency:`t%Transparent%`nTransColor:`t%TransC
olor%
return
```

```
; Examples of "WinSet Region":
```

```
WinSetRegion, 50-0 W200 H250, WinTitle ; Make all
parts of the window outside this rectangle
invisible.
```

```
WinSetRegion, 50-0 W200 H250 R40-40, WinTitle ;
Same as above but with corners rounded to 40x40.
```

```
WinSetRegion, 50-0 W200 H250 E, WinTitle ; An
ellipse instead of a rectangle.
```

```
WinSetRegion, 50-0 250-0 150-250, WinTitle ; A
triangle with apex pointing down.
```

```
WinSetRegion,, WinTitle ; Restore the window to
its original/default display area.
```

```
; Here is a region with a more complex area. It
creates a see-through rectangular hole inside a
window.
```

```
; There are two rectangles specified below: an
outer and an inner. Each rectangle consists of 5
pairs
```

```
; of X/Y coordinates because the first pair is
repeated at the end to "close off" each rectangle.
```

```
WinSetRegion, 0-0 300-0 300-300 0-300 0-0 100-
100 200-100 200-200 100-200 100-100, WinTitle
```

# WinSetTitle

Changes the title of the specified window.

```
WinSetTitle NewTitle [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetTitle "New Title", "A"
Function Example: WinSetTitle("New Title", "A")
```

## Parameters

### NewTitle

The new title for the window. If this is the only parameter given, the [Last Found Window](#) will be used.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included [Window Spy](#) utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

A change to a window's title might be merely temporary if the application that owns the window frequently changes the title.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinMove](#), [WinGetTitle](#), [WinGetText](#), [ControlGetText](#), [WinGetPos](#)

## Example

```
WinSetTitle("This is a new title", "Untitled - Notepad")
```

**; Alternate:**

```
Run("notepad.exe")
WinWaitActive("Untitled - Notepad")
WinSetTitle("This is a new title") ; Uses the
window found above by WinWaitActive
```

# WinShow

Unhides the specified window.

```
WinShow [WinTitle, WinText, ExcludeTitle, ExcludeText]
```

```
Command Example: WinShow "ahk_class
AutoHotkeyGUI"
```

```
Function Example: WinShow("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

By default, WinShow is the only command that can always detect hidden windows. Other commands can detect them only if [DetectHiddenWindows](#) has been turned on.

This command operates only upon the topmost matching window except when *WinTitle* is `ahk_group GroupName`, in which case all windows in the group are affected.

## Related

[WinHide](#), [SetTitleMatchMode](#), [DetectHiddenWindows](#), [Last Found Window](#)

## Example

```
Run, notepad.exe
WinWait, Untitled - Notepad
Sleep, 500
WinHide ; Because its parameter is omitted, it
uses the window found above.
Sleep, 1000
WinShow
```

# WinWait

Waits until the specified window exists.

```
WinWait [WinTitle, WinText, Seconds, ExcludeTitle, ExcludeText]
```

```
Command Example: WinWait "ahk_class
AutoHotkeyGUI"
```

```
Function Example: WinWait("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

*WinTitle* may be blank only when *WinText*, *ExcludeTitle*, or *ExcludeText* is present.

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### Seconds

How many seconds to wait before timing out and setting `ErrorLevel` to 1. Leave blank to wait indefinitely. Specifying 0 is the same as specifying 0.5.

### **ExcludeTitle**

Windows whose titles include this value will not be considered.

### **ExcludeText**

Windows whose text include this value will not be considered.

## **ErrorLevel**

`ErrorLevel` is set to 1 if the command timed out or 0 otherwise.

## **Remarks**

If a matching window comes into existence, the command will not wait for `Seconds` to expire. Instead, it will immediately set `ErrorLevel` to 0, update the `Last Found Window`, and the script will continue executing.

While the command is in a waiting state, new `threads` can be launched via `hotkey`, `custom menu item`, or `timer`.

If another `thread` changes the contents of any variable(s) that were used for this command's parameters, the command will not see the change -- it will continue to use the title and text that were originally present in the variables when the command first started waiting.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinWaitActive](#), [WinWaitClose](#), [WinExist](#), [WinActive](#), [ProcessExist](#), [SetTitleMatchMode](#), [DetectHiddenWindows](#)

## Example

```
Run, notepad.exe
WinWait, Untitled - Notepad, , 3
if ErrorLevel
{
 MsgBox, WinWait timed out.
 return
}
else
 WinMinimize ; Minimize the window found by
WinWait.
```

# WinWaitActive / WinWaitNotActive

Waits until the specified window is active or not active.

```
WinWaitActive [WinTitle, WinText, Seconds,
ExcludeTitle, ExcludeText]
WinWaitNotActive [, WinTitle, WinText, Seconds,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinWaitActive "ahk_class
AutoHotkeyGUI"
```

```
Function Example: WinWaitActive("ahk_class
AutoHotkeyGUI")
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### Seconds

How many seconds to wait before timing out and setting [ErrorLevel](#) to 1.

Leave blank to wait indefinitely. Specifying 0 is the same as specifying 0.5.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if the command timed out or 0 otherwise.

## Remarks

If a matching window satisfies the command's expectation, the command will not wait for *Seconds* to expire. Instead, it will immediately set `ErrorLevel` to 0 and the script will continue executing.

Both `WinWaitActive` and `WinWaitNotActive` will update the `Last Found Window` if a qualified window is active when the command begins. In addition, `WinWaitActive` will update the `Last Found Window` if a qualified window becomes active before the command times out.

While the command is in a waiting state, new `threads` can be launched via `hotkey`, `custom menu item`, or `timer`.

If another `thread` changes the contents of any variable(s) that were used for this

command's parameters, the command will not see the change -- it will continue to use the title and text that were originally present in the variables when the command first started waiting.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[WinWait](#), [WinWaitClose](#), [WinExist](#), [WinActive](#), [SetTitleMatchMode](#), [DetectHiddenWindows](#)

## Example

```
Run, notepad.exe
WinWaitActive, Untitled - Notepad, , 2
if ErrorLevel
{
 MsgBox, WinWait timed out.
 return
}
else
 WinMinimize ; minimize the window found by
WinWaitActive.
```

# WinWaitClose

Waits until the specified window does not exist.

```
WinWaitClose [WinTitle, WinText, Seconds,
ExcludeTitle, ExcludeText]
```

```
Command example: WinWaitClose "MyGui ahk_class
AutoHotkeyGUI",, 0.2
```

```
Function example: WinWaitClose("MyGui ahk_class
AutoHotkeyGUI",, 0.2)
```

## Parameters

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### Seconds

How many seconds to wait before timing out and setting [ErrorLevel](#) to 1. Leave blank to wait indefinitely. Specifying 0 is the same as specifying 0.5.

## ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## ErrorLevel

`ErrorLevel` is set to 1 if the command timed out or 0 otherwise.

## Remarks

Whenever no instances of the specified window exist, the command will not wait for *Seconds* to expire. Instead, it will immediately set `ErrorLevel` to 0 and the script will continue executing.

While the command is in a waiting state, new `threads` can be launched via `hotkey`, `custom menu item`, or `timer`.

If another `thread` changes the contents of any variable(s) that were used for this command's parameters, the command will not see the change -- it will continue to use the title and text that were originally present in the variables when the command first started waiting.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

WinClose, WinWait, WinWaitActive, WinExist, WinActive, ProcessExist,  
SetTitleMatchMode, DetectHiddenWindows

## Example

```
Run, notepad.exe
WinWait, Untitled - Notepad
WinWaitClose ; Wait for the exact window found by
WinWait to be closed.
MsgBox, Notepad is now closed.
```

# UnZip

Extract one or all items from a zip archive.

```
OutputVar := UnZip(ZipFileName, DestinationFolder [,
FileToExtract, DestinationFileName, Password])
```

```
OutputVar := UnZip(Address, Size, DestinationFolder
[, FileToExtract, DestinationFileName, Password])
```

```
Command Example: UnZip "MyZip.zip", "C:\Temp",
"scripts\MyScript.ahk"
```

```
UnZip &zip;, sz, "C:\Temp",
"scripts\MyScript.ahk"
```

```
Function Example: Success := UnZip("MyZip.zip",
"C:\Temp", "scripts\MyScript.ahk")
```

```
Success := UnZip(&zip;, sz,
"C:\Temp", "scripts\MyScript.ahk")
```

## Parameters

### OutputVar

The name of the variable in which to store 1 / true if file(s) was extracted successfully or 0 / false if operation failed.

### ZipFileName or Address

The path and name of zip archive or an address to memory containing a zip archive.

## Size

When previous parameter is an address, the size of zip archive in memory otherwise this parameter is skipped, see examples.

## DestinationFolder

Path where to extract the file(s) to.

## FileToExtract (optional)

Relative path and name or zero based index of the file within zip archive that will be extracted. If omitted all files will be extracted.

## DestinationFileName (optional)

FileName for the file that will be extracted. If omitted original file name from zip archive will be used.

## Password (optional)

Password for zip archive.

## Related

[ZipCreateFile](#), [ZipAddFile](#), [ZipCloseFile](#), [ZipOptions](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#), [UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
UnZip("C:\Test.zip", "C:\Temp\Test")
```

# UnZipBuffer

Extract one items from a zip archive.

```
OutputVar := UnZipBuffer(ZipFileName, FileToExtract
[, VariableName, Password])
```

```
OutputVar := UnZipBuffer(Address, Size, FileToExtract
[, VariableName, Password])
```

```
Command Example: UnZipBuffer "MyZip.zip",
"scripts\MyScript.ahk", var
UnZipBuffer &zip;, sz,
"scripts\MyScript.ahk", var
```

```
Function Example: Success :=
UnZipBuffer("MyZip.zip",
"scripts\MyScript.ahk", var)
Success := UnZipBuffer
(&zip;, sz, "scripts\MyScript.ahk", var)
```

## Parameters

### OutputVar

The name of the variable in which to store the size of extracted file.

### ZipFileName or Address

The path and name of zip archive to extract from or an address to memory containing a zip archive.

### Size

The size in bytes of zip archive in memory. When previous parameter is ZipFileName this parameter is skipped, see examples.

### FileToExtract

Relative path and name or zero based index of the file within zip archive that will be extracted.

### VariableName (optional)

The name of the variable in which to store extracted file.

### Password (optional)

Password for zip archive.

## Related

[ZipCreateFile](#), [ZipAddFile](#), [ZipCloseFile](#), [ZipOptions](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#), [UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
sz := UnZipBuffer("C:\Test.zip", "Test.txt", var)
```

# UnZipRawMemory

This function is used to unzip and decrypt raw memory, for example from resource.

```
OutputVar := UnZipRawMemory(Address, Size, VariableOrAddress, Password)
```

```
Command Example: UnZipRawMemory &resource;,
SizeOfResource(NULL,hRes), Data
Function Example: Size :=
UnZipRawMemory(&resource;,
SizeOfRes(NULL,hRes), Data)
```

## Parameters

### OutputVar

The name of the variable in which to store the size of unzipped memory in bytes.

### Address

Address to raw zipped memory.

### Size

Size in bytes of zipped memory.

### VariableOrAddress (optional)

The name of a variable or address to copy unzipped memory to.

### Password (optional)

The password to use to decrypt the data.

## Remarks

Ahk2Exe uses [ZipRawMemory](#) to compress and encrypt resources in raw mode (only 1 file supported).

Internally AutoHotkey is able to decompress raw resources automatically (compiled script, AutoHotkey.dll, WinApi + lib resources). This function can be used in script to decrypt and decompress raw resources.

## Related

[ZipRawMemory](#), [ZipCreateFile](#), [ZipAddFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
sz:=ResGet(data, A_AhkPath, "AHKEXEC.AHK", "LIB")
UnZipRawMemory(&data;, sz, var)
MsgBox % StrGet(&var;, "UTF-8")
```

# ZipAddBuffer

Add a file from memory to a zip archive created with [ZipCreateBuffer](#) or [ZipCreateFile](#).

```
OutputVar := ZipAddBuffer(ZipHandle, Address, Size, FileName)
```

```
Command Example: ZipAddBuffer hZip, &var;, sz, "MyScript.ahk"
```

```
Function Example: Success := ZipAddBuffer(hZip, &var;, sz, "MyScript.ahk")
```

## Parameters

### OutputVar

The name of the variable in which to store 1 / true if file was added successfully or 0 / false if operation failed.

### ZipHandle

Zip handle returned from [ZipCreateFile](#) or [ZipCreateBuffer](#).

### Address

Address of data to be added to zip archive.

### Size

Size in bytes of data in memory.

## FileName

Name of the file that will be added to zip archive.

## Related

[ZipCreateFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#), [ZipAddFile](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#), [UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
hZip:=ZipCreateBuffer(10000000)
ZipAddBuffer(hZip, "C:\MyScript.ahk", &var;, sz)
sz := ZipCloseBuffer(hZip, var)
```

# ZipAddFile

Add a file to a zip archive created with [ZipCreateFile](#) or [ZipCreateBuffer](#).

```
OutputVar := ZipAddFile(ZipHandle, FileName [, ZipFileName])
```

```
Command Example: ZipAddFile hZip,
"MyScript.ahk", "scripts\MyScript.ahk"
Function Example: Success := ZipAddFile(hZip,
"MyScript.ahk", "scripts\MyScript.ahk")
```

## Parameters

### OutputVar

The name of the variable in which to store 1 / true if file was added successfully or 0 / false if operation failed.

### ZipHandle

Zip handle returned from [ZipCreateFile](#) or [ZipCreateBuffer](#).

### FileName

Name or path of the file that will be added to zip archive.

### ZipFileName (optional)

Path and name of the file to use for zip archive. If omitted original file name will be used.

## Related

[ZipCreateFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#),  
[ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#),  
[UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
hZip:=ZipCreateFile("C:\Test.zip")
ZipAddFile(hZip, "C:\MyScript.ahk")
ZipCloseFile(hZip)
```

# ZipAddFolder

Add an empty folder to zip archive created with [ZipCreateFile](#) or [ZipCreateBuffer](#).

```
OutputVar := ZipAddFolder(ZipHandle, ZipFileName)
```

```
Command Example: ZipAddFolder hZip, "scripts"
Function Example: Success := ZipAddFolder(hZip,
"scripts")
```

## Parameters

### OutputVar

The name of the variable in which to store 1 / true if file was added successfully or 0 / false if operation failed.

### ZipHandle

Zip handle returned from [ZipCreateFile](#) or [ZipCreateBuffer](#).

### ZipFileName

Path and name of the folder to use for zip archive.

## Related

[ZipCreateFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#),  
[ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#),

UnZipRawMemory, ZipInfo, ZipAddFile

## Examples

```
hZip:=ZipCreateFile("C:\Test.zip")
ZipAddFolder(hZip, "scripts")
ZipCloseFile(hZip)
```

# ZipCloseBuffer

Closes zip archive created with [ZipCreateBuffer](#), saves it to variable and returns its size.

```
OutputVar := ZipCloseBuffer(ZipHandle |, |
VariableName)
```

```
Function Example: Size := ZipCloseBuffer(hZip,
var)
```

## Parameters

### OutputVar

The name of the variable in which to store the size in bytes of zip file in memory.

### ZipHandle

Zip handle returned from [ZipCreateBuffer](#).

### VariableName

The name of the variable in which to store the zip archive.

## Related

[ZipCreateFile](#), [ZipAddFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#), [UnZipRawMemory](#), [ZipInfo](#),

## ZipAddFolder

### Examples

```
hZip:=ZipCreateBuffer(10000000)
ZipAddBuffer(hZip, "C:\MyScript.ahk", &var;, sz)
sz := ZipCloseBuffer(hZip, var)
```

# ZipCloseFile

Closes zip archive created with [ZipCreateFile](#).

```
OutputVar := ZipCloseFile(ZipHandle)
```

```
Command Example: ZipCloseFile hZip
Function Example: Success := ZipCloseFile(hZip)
```

## Parameters

### OutputVar

The name of the variable in which to store 1 / true if file was added successfully or 0 / false if operation failed.

### ZipHandle

Zip handle returned from [ZipCreateFile](#).

## Related

[ZipCreateFile](#), [ZipAddFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#), [UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
hZip:=ZipCreateFile("C:\Test.zip")
```



# ZipCreateBuffer

This function is used to create a new empty zip file in memory, use [ZipAddBuffer](#) or [ZipAddFile](#) to add files to the zip archive.

```
OutputVar := ZipCreateBuffer(MaxSize [, Password])
```

```
Function Example: hZip :=
ZipCreateBuffer(10000000)
```

## Parameters

### OutputVar

The name of the variable in which to store the zip handle that can be used in [ZipAddBuffer](#), [ZipAddFile](#) and [ZipCloseBuffer](#).

### MaxSize

Maximum size for zip archive in memory, you can make this a very large number because memory will be only reserved, but not actually committed, until needed.

### Password (optional)

The password to use for the zip file.

## Remarks

Use `ZipCreateBuffer` to create a new zip file, then use `ZipAddFile` or `ZipAddBuffer` to add files to it and finally use `ZipCloseBuffer` to close the file. You cannot add files to existing zip file or after `ZipCloseBuffer` has been called, instead `UnZip` all files and create a new zip file.

A `_ZipCompressionLevel` can be used to change compression level. Use 0 for lowest and 9 for highest compression.

## Related

`ZipCreateFile`, `ZipAddFile`, `ZipCloseFile`, `ZipOptions`, `UnZip`, `ZipAddBuffer`, `ZipCloseBuffer`, `UnZipBuffer`, `ZipRawMemory`, `UnZipRawMemory`, `ZipInfo`, `ZipAddFolder`

## Examples

```
hZip:=ZipCreateBuffer(10000000)
ZipAddFile(hZip, "C:\MyScript.ahk")
sz := ZipCloseBuffer(hZip, var)
```

# ZipCreateFile

This function is used to create a new empty zip file, use [ZipAddFile](#) or [ZipAddBuffer](#) to add files to the zip archive.

```
OutputVar := ZipCreateFile(FileName [, Password])
```

```
Function Example: hZip :=
ZipCreateFile("MyScript.zip", "MyPassword")
```

## Parameters

### OutputVar

The name of the variable in which to store the zip handle that can be used in [ZipAddFile](#), [ZipAddBuffer](#) and [ZipCloseFile](#).

### FileName

The name or path of the zip file to be created.

### Password (optional)

The password to use for the zip file.

## Remarks

Use [ZipCreateFile](#) to create a new zip file, then use [ZipAddFile](#) or [ZipAddBuffer](#) to add files to it and finally use [ZipCloseFile](#) to close the file.

You cannot add files to existing zip file or after ZipCloseFile has been called, instead UnZip all files and create a new zip file.

A\_ZipCompressionLevel can be used to change compression level. Use 0 for lowest and 9 for highest compression.

## Related

[ZipAddFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#), [UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
hZip:=ZipCreateFile("C:\Test.zip")
ZipAddFile(hZip, "C:\MyScript.ahk")
ZipCloseFile(hZip)
```

# ZipInfo

Return an object with information about all items within a zip archive.

```
OutputVar := ZipInfo(FileNameOrAddress [, Size])
```

```
Function Example: obj :=
ZipInfo("MyZipFile.zip")
```

## Parameters

### OutputVar

The name of the variable in which to store the object containing information about all items.

Each item contains an object with following information: **FileName**, **Attributes**, **CompressedSize**, **UncompressedSize**, **CreateTime**, **ModifyTime**, **AccessTime**.

### FileNameOrAddress

FileName of zip file or address of zip file in Memory.

### Size (optional)

When FileNameOrAddress is an address, this parameter must be the size of zip memory returned by [ZipCloseBuffer](#).

## Related

ZipCreateFile, ZipAddFile, ZipCloseFile, UnZip, ZipCreateBuffer,  
ZipAddBuffer, ZipCloseBuffer, UnZipBuffer, ZipRawMemory,  
UnZipRawMemory, ZipOptions, ZipAddFolder

## Examples

```
Gui, Add, ListView, w800
h640, Name|Attributes|CompressedSize|UncompressedSi
ze|CreateTime|ModifyTime|AccessTime
for k, v in ZipInfo("C:\Test.zip")
 LV_Add("", v.Name, v.Attributes,
v.CompressedSize, v.UncompressedSize,
v.CreateTime, v.ModifyTime, v.AccessTime)
LV_ModifyCol()
Gui, Show
return
GuiClose:
ExitApp
```

# ZipOptions

Changes options for zip archive created with [ZipCreateFile](#).

```
OutputVar := ZipOptions(ZipHandle, Options)
```

```
Command Example: ZipOptions hZip, 0x80000000
Function Example: Success := ZipOptions(hZip,
0x80000000)
```

## Parameters

### OutputVar

The name of the variable in which to store 1 / true if file was added successfully or 0 / false if operation failed.

### ZipHandle

Zip handle returned from [ZipCreateFile](#).

### Options

The only supported options is TZIP\_OPTION\_GZIP = 0x80000000.

## Remarks

When TZIP\_OPTION\_GZIP is used only one file can be included in the zip archive.

## Related

[ZipCreateFile](#), [ZipAddFile](#), [ZipCloseFile](#), [UnZip](#), [ZipCreateBuffer](#),  
[ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipRawMemory](#),  
[UnZipRawMemory](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
hZip:=ZipCreateFile("C:\Test.zip")
ZipOptions(hZip, 0x80000000)
ZipAddFile(hZip, "C:\MyScript.ahk")
ZipCloseFile(hZip)
```

# ZipRawMemory

This function is used to zip and decrypt raw memory, for example to use for resources.

```
OutputVar := ZipRawMemory(Address, Size, [,
VariableOrAddress, Password])
```

```
Function Example: Size := ZipRawMemory(&var,,
sizeofvar, Data)
```

## Parameters

### OutputVar

The name of the variable in which to store the size of zipped memory in bytes.

### Address

Address to memory to be zipped.

### Size

Size in bytes of data in memory.

### VariableOrAddress (optional)

The name of a variable or address to copy zipped memory to. If omitted only the size will be returned.

## Password (optional)

The password to use to encrypt the data.

## Remarks

Ahk2Exe uses ZipRawMemory to compress and encrypt resources in raw mode (only 1 file supported).

Internally AutoHotkey is able to decompress raw resources automatically (compiled script, AutoHotkey.dll, WinAPI + lib resources). [UnZipRawMemory](#) can be used in script to decrypt and decompress data compressed with ZipRawMemory.

## Related

[ZipRawMemory](#), [ZipCreateFile](#), [ZipAddFile](#), [ZipCloseFile](#), [ZipOptions](#), [UnZip](#), [ZipCreateBuffer](#), [ZipAddBuffer](#), [ZipCloseBuffer](#), [UnZipBuffer](#), [ZipInfo](#), [ZipAddFolder](#)

## Examples

```
string:="This is a message from AutoHotkey."
sz := ZipRawMemory(string, StrLen(string) * 2,
var, "password")
UnZipRawMemory(&var,, sz, text, "password")
MsgBox % text
```

# #ClipboardTimeout

Changes how long the script keeps trying to access the clipboard when the first attempt fails.

**#ClipboardTimeout** Milliseconds

## Parameters

### Milliseconds

The length of the interval in milliseconds. Specify -1 to have it keep trying indefinitely. Specify 0 to have it try only once. Scripts that do not contain this directive use a 1000 ms timeout.

## Remarks

Some applications keep the clipboard open for long periods of time, perhaps to write or read large amounts of data. In such cases, increasing this setting causes the script to wait longer before giving up and displaying an error message.

This settings applies to all [clipboard](#) operations, the simplest of which are the following examples: `Var := Clipboard` and `Clipboard := "New Text"`.

Whenever the script is waiting for the clipboard to become available, new [threads](#) cannot be launched and [timers](#) will not run. However, if the user presses a [hotkey](#), selects a [custom menu item](#), or performs a [GUI action](#) such as pressing

a button, that event will be buffered until later; in other words, its subroutine will be performed after the clipboard finally becomes available.

This directive does **not** cause the reading of clipboard data to be reattempted if the first attempt fails. Prior to v1.1.16, it did cause the script to wait until the timeout expired, but in doing so prevented any further data from being retrieved.

## Related

[Clipboard](#), [Thread](#)

## Example

```
#ClipboardTimeout 2000
```

# #DllImport

Creates a script function for a dll or exe function.

```
#DllImport Function_Name, [DllFile\]Function [, Type1,
Arg1, Type2, Arg2, Cdecl ReturnType]
```

```
Example: #DllImport, ExecScript,
%A_AhkPath%\ahkExec,Str,,Cdecl
```

## Parameters

### Function\_Name

The name of new function. See example.

### [DllFile\]Function

The DLL or EXE file name followed by a backslash and the name of the function. For example: `"MyDLL\MyFunction"` (the file extension ".dll" is the default when omitted). If an absolute path isn't specified, *DllFile* is assumed to be in the system's PATH or [A\\_WorkingDir](#).

Following built-in variables can be used too: `A_ScriptDir`, `A_AhkPath`, `A_DllPath`, `A_AppData`, `A_AppDataCommon`.

*DllFile* may be omitted when calling a function that resides in `User32.dll`, `Kernel32.dll`, `ComCtl32.dll`, or `Gdi32.dll`. For example,

`"User32\IsWindowVisible"` produces the same result as

`"IsWindowVisible"`.

If no function can be found by the given name, a "W" (Unicode) suffix is automatically appended. For example, `"MessageBox"` is the same as `"MessageBoxW"`.

This parameter may also consist solely of an integer, which is interpreted as the address of the function to call. Sources of such addresses include [COM](#) and [RegisterCallback](#).

**Note:** `9MyFunction` or `0MyDll.dll\Function` are not supported because they will be assumed integers and will cause an exception to be thrown.

This parameter may also be a hex string that represents the code. Hex code for 64-bit can be appended after 32-bit code separated using colon (:): `[32-bit hex]:[64-bit hex]`. See also examples below.

### Type1, Arg1 (optional)

Each of these pairs represents a single parameter to be passed to the function. The number of pairs is unlimited. For *Type*, see the [types table](#). *Type* can be specified without quotes.

For *Arg*, specify the default value to be passed to the function. This parameters will be only used if corresponding parameter is omitted by function call. If *Arg* is a string it must not be enclosed in quotes (""") and all parameters starting with integer will be converted to digits, hexadecimal values are supported too.

### Cdecl ReturnType (optional)

The word *Cdecl* is normally omitted because most functions use the standard calling convention rather than the "C" calling convention (functions such as `wsprintf` that accept a varying number of arguments are one exception to this). Note that most object-oriented C++ functions use the *thiscall* convention, which is not supported.

If present, the word *Cdecl* should be listed before the return type (if any). Separate each word from the next with a space or tab. For example:

```
"Cdecl Str".
```

Since a separate "C" calling convention does not exist in 64-bit code, *Cdecl* may be specified but has no effect on 64-bit builds of AutoHotkey.

*ReturnType*: If the function returns a 32-bit signed integer (`Int`), `BOOL`, or nothing at all, *ReturnType* may be omitted. Otherwise, specify one of the argument types from the [types table](#) below. The [asterisk suffix](#) is also supported.

## General Remarks

String parameters for `Arg` must not be enclosed in quotes (""). Parameters starting with number are assumed numbers, so for example `0ABC` will be considered 0. Hexadecimal values are allowed too, so `0xA` will be converted to 10.

## Related

[DllCall](#), [DynaCall](#), [WinApi](#)

## Examples

```
#DllImport,
SendMsg,user32\SendMessage,PTR,0,UInt,0x999,PTR,11
1,PTR,222,PTR
OnMessage(0x999,"0x999")
```

```
;
```

```
DllCall("SendMessage","PTR",A_ScriptHwnd,"UINT",0x
999,"PTR",111,"PTR",222,"PTR")
```

```
SendMsg(A_ScriptHwnd)
SendMsg,,%A_ScriptHwnd%
```

```
;
```

```
DllCall("SendMessage","PTR",A_ScriptHwnd,"UINT",0x
999,"PTR",1,"PTR",2,"PTR")
```

```
SendMsg(A_ScriptHwnd,,1,2)
SendMsg,,%A_ScriptHwnd%,,1,2
ExitApp
```

```
0x999(w:=0,l:=0,m:=0,h:=0){
 MsgBox % w "`n" l "`n" m "`n" h
 return 1
}
```

```
; Import RGB_TO_BGR function as hex (separated by
:), first hex is 32-bit and second hex 64-bit
version.
```

```
; DWORD RGB_TO_BGR(COLORREF rgb){
; return ((rgb & 255) << 16) | (((rgb >> 8)
& 255) << 8) | (rgb >> 16);
; }
```

```
#DllImport,RGB_TO_BGR,8B4C24040FB6C18BD1C1E01081E2
00FF00000BC2C1E9100BC1C3:0FB6C18BD1C1E910C1E01081E
200FF00000BC20BC1C3,UInt,,CDecl UInt
MsgBox % format("0x{1:X}",RGB_TO_BGR(0xAABBCC))
```



# #ErrorStdOut

Sends any syntax error that prevents a script from launching to **stderr** rather than displaying a dialog.

## #ErrorStdOut

Errors are written to stderr instead of stdout. The command prompt and fancy editors usually display both. This change was undocumented until after v1.1.19.01.

This allows fancy editors such as Textpad, SciTE, Crimson, and EditPlus to jump to the offending line when a syntax error occurs. Since the #ErrorStdOut directive would have to be added to every script, it is usually better to set up your editor to use the [command line switch](#) /ErrorStdOut when launching any AutoHotkey script (see further below for setup instructions).

Because AutoHotkey is not a console program, errors will not appear at the command prompt directly. Instead, such output can be captured via piping or redirection. For example:

```
"C:\Program Files\AutoHotkey\AutoHotkey.exe"
/ErrorStdOut "My Script.ahk" 2>&1 |more
"C:\Program Files\AutoHotkey\AutoHotkey.exe"
/ErrorStdOut "My Script.ahk" 2>"Syntax-Error
Log.txt"
```

You can also pipe the output directly to the clipboard by downloading [cb.zip](#) (4

KB) and then following this example:

```
"C:\Program Files\AutoHotkey\AutoHotkey.exe"
/ErrorStdOut "My Script.ahk" 2>&1 | cb.exe
```

**Note:** `2>&1` causes stderr to be redirected to stdout, while `2>Filename` redirects only stderr to a file.

## Instructions for specific editors

### EditPlus:

From the menu bar, select Tools > Configure User Tools.

Press button: Add Tool > Program

Menu Text: Your choice

Command: C:\Program Files\AutoHotkey\AutoHotkey.exe

Argument: /ErrorStdOut "\$(FilePath)"

Initial directory: \$(FileDir)

Capture output: Yes

### TextPad:

From the menu bar, select Configure > Preferences.

Expand the Tools entry.

Press the Add button and select "Program".

Copy and paste (adjust to your path): *C:\Windows\System32\cmd.exe* -- then press OK.

Triple-click the newly added item (cmd.exe) in the ListBox and rename it to your choice (e.g. Launch Script).

Press Apply.

Select the new item in the tree at the left and enter the following information:

Command (should already be filled in): cmd.exe (or the full path to it)

Parameters (adjust to your path, if necessary): /c ""C:\Program Files\AutoHotkey\AutoHotkey.exe" /ErrorStdOut "\$File""

Initial folder: \$FileDir

Check the following boxes: 1) Run minimized; 2) Capture output.

Press OK. The newly added item should now exist in the Tools menu.

## Related

[FileAppend](#) (because it can also send text to stdout)

## Example

```
#ErrorStdOut
```

# #NoTrayIcon

Disables the showing of a tray icon.

```
#NoTrayIcon
```

Specifying this anywhere in a script will prevent the showing of a tray icon for that script when it is launched (even if the script is compiled into an EXE).

If you use this for a script that has hotkeys, you might want to bind a hotkey to the [ExitApp](#) command. Otherwise, there will be no easy way to exit the program (without restarting the computer or killing the process). For example:

```
#x::ExitApp.
```

The tray icon can be made to disappear or reappear at any time during the execution of the script by using the command `Menu, Tray, Icon` or `Menu, Tray, NoIcon`. The only drawback of using `Menu, Tray, NoIcon` at the very top of the script is that the tray icon might be briefly visible when the script is first launched. To avoid that, use `#NoTrayIcon` instead.

The built-in variable `A_IconHidden` contains 1 if the tray icon is currently hidden or 0 otherwise.

## Related

[Menu](#), [ExitApp](#)

## Example

```
#NoTrayIcon
```

# #Persistent

Keeps a script permanently running (that is, until the user closes it or `ExitApp` is encountered).

## #Persistent

If this directive is present anywhere in the script, that script will stay running after the auto-execute section (top part of the script) completes and all other threads have exited. This is usually unnecessary because the script automatically becomes persistent when the program detects that it has some way of launching new threads, such as a hotkey, timer or GUI.

Some cases where this directive might be needed (if there are no running threads or hotkeys, timers, etc.) include:

- Scripts which use `RegisterCallback` and `DllCall` to respond to events, since those functions don't make the script persistent.
- Scripts which create or retrieve COM objects and use `ComObjConnect` to respond to the object's events.

If this directive is added to an existing script, you might want to change some or all occurrences of `Exit` to be `ExitApp`. This is because `Exit` will not terminate a persistent script; it terminates only the current thread.

## Related

Exit, ExitApp

## Example

```
; This script will not exit automatically, even
though it has nothing to do.
; However, you can use its tray icon to open the
script in an editor, or to
; launch Window Spy or the Help file.
#Persistent
```

# #SingleInstance

Determines whether a script is allowed to run again when it is already running.

```
#SingleInstance [force|ignore|off]
```

## Parameters

`force|ignore|off`

This parameter determines what happens when a script is launched while a previous instance of itself is already running. If specified, it must be one of the following words:

**Force:** Skips the dialog box and replaces the old instance automatically, which is similar in effect to the [Reload](#) command.

**Ignore:** Skips the dialog box and leaves the old instance running. In other words, attempts to launch an already-running script are ignored.

**Off:** Allows multiple instances of the script to run concurrently.

If the parameter is omitted, it defaults to *Force*.

## Remarks

If this directive is not used, a dialog box is displayed asking whether to keep the old instance or replace it with the new one.

This directive is ignored when any of the following [command line switches](#) are used: /force /f /restart /r

## Related

[Reload](#)

## Example

```
#SingleInstance force
#SingleInstance ignore
#SingleInstance off
```

# #Warn

Enables or disables warnings for specific conditions which may indicate an error, such as a typo or missing "global" declaration.

```
#Warn [WarningType, WarningMode]
```

## Parameters

### WarningType

The type of warning to enable or disable. If omitted, it defaults to *All*.

**UseUnsetLocal** or **UseUnsetGlobal**: Warn when a variable is read without having previously been assigned a value or initialized with [VarSetCapacity](#). If the variable is intended to be empty, assign an empty string to suppress this warning.

This is split into separate warning types for locals and globals because it is more common to use a global variable without prior initialization, due to their persistent and script-wide nature. For this reason, some script authors may wish to enable this type of warning for locals but disable it for globals.

```
#Warn
;y := "" ; This would suppress the
warning.
x := y ; y hasn't been assigned a value.
```

**LocalSameAsGlobal:** Before the script starts to run, display a warning for each *undeclared* local variable which has the same name as a global variable. This is intended to prevent errors caused by forgetting to declare a global variable inside a function before attempting to access it. If the variable really was intended to be local, a declaration such as `local x` or `static y` can be used to suppress the warning.

```
#Warn
g := 1
ShowG() {
 ; The warning is displayed
 ; even if the function is never called.
 ;global g ; <-- This is required to
 ; access the global variable.
 MsgBox % g ; Without the declaration,
 "g" is an empty local variable.
}
```

**All:** Apply the given *WarningMode* to all supported warning types.

## WarningMode

A value indicating how warnings should be delivered. If omitted, it defaults to *MsgBox*.

**MsgBox:** Show a message box describing the warning. Note that once the message box is dismissed, the script will continue as usual.

**StdOut:** Send a description of the warning to *stdout* (the program's standard output stream), along with the filename and line number. This allows fancy editors such as SciTE to capture warnings without

disrupting the script - the user can later jump to each offending line via the editor's output pane.

**OutputDebug:** Send a description of the warning to the debugger for display. If a debugger is not active, this will have no effect. For more details, see [OutputDebug](#).

**Off:** Disable warnings of the given *WarningType*.

## Remarks

By default, all warnings are off.

Warnings can't be enabled or disabled at run-time; the settings are determined when a script loads. Therefore, the location in the script is not significant (and, like other `#` directives, `#Warn` cannot be executed conditionally).

However, the ordering of multiple `#Warn` directives is significant: the last occurrence that sets a given warning determines the mode for that warning. So, for example, the two statements below have the combined effect of enabling all warnings except `UseUnsetGlobal`:

```
#Warn All
#Warn UseUnsetGlobal, Off
```

## Related

[Local and Global Variables](#)

## Example

```
#Warn All, Off ; Disable all warnings. This is the default state.
#Warn ; Enable every type of warning; show each warning in a message box.
#Warn UseUnsetLocal, OutputDebug ; Warn when a local variable is used before it's set; send warning to OutputDebug.
```

# #WarnContinuableException

Does not display the EXCEPTION\_ACCESS\_VIOLATION warning message for continuable exceptions.

**#WarnContinuableException** [On|Off]

## Parameters

### On|Off

#WarnContinuableException without one of the following words after it is equivalent to #WarnContinuableException On.

**On (default):** The EXCEPTION\_ACCESS\_VIOLATION warning message will be shown for continuable and noncontinuable exception.

**Off:** The EXCEPTION\_ACCESS\_VIOLATION warning message will be shown for noncontinuable exception only.

# Software License

## GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

```
Copyright (C) 1989, 1991 Free Software
Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA
02110-1301, USA
```

```
Everyone is permitted to copy and distribute
verbatim copies
of this license document, but changing it is
not allowed.
```

## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to

distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain

patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## **TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and

appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a

copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

**3.** You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

**a)** Accompany it with the complete corresponding machine-readable

source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

**b)** Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

**c)** Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code

from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on

you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

**8.** If the distribution and/or use of the Program is restricted in certain countries

either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

**9.** The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

**10.** If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**END OF TERMS AND CONDITIONS**

# Regular Expressions (RegEx) - Quick Reference

## Fundamentals

**Match anywhere:** By default, a regular expression matches a substring *anywhere* inside the string to be searched. For example, the regular expression `abc` matches `abc123`, `123abc`, and `123abcxyz`. To require the match to occur only at the beginning or end, use an [anchor](#).

**Escaped characters:** Most characters like `abc123` can be used literally inside a regular expression. However, the characters `\.*?+[]{}()^$` must be preceded by a backslash to be seen as literal. For example, `\.` is a literal period and `\\` is a literal backslash. Escaping can be avoided by using `\Q...\E`. For example: `\QLiteral Text\E`.

**Case-sensitive:** By default, regular expressions are case-sensitive. This can be changed via the `"i"` option. For example, the pattern `()abc` searches for "abc" without regard to case. See below for other modifiers.

## Options (case sensitive)

At the very beginning of a regular expression, specify zero or more of the following options followed by a close-parenthesis. For example, the pattern "**im**)abc" would search for *abc* with the case-insensitive and multiline options (the parenthesis may be omitted when there are no options). Although this syntax breaks from tradition, it requires no special delimiters (such as forward-slash), and thus there is no need to escape such delimiters inside the pattern. In addition, performance is improved because the options are easier to parse.

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>i</b> | Case-insensitive matching, which treats the letters A through Z as identical to their lowercase counterparts.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>m</b> | <p>Multiline. Views <i>Haystack</i> as a collection of individual lines (if it contains newlines) rather than as a single continuous line. Specifically, it changes the following:</p> <ol style="list-style-type: none"><li>1) Circumflex (^) matches immediately after all internal newlines -- as well as at the start of <i>Haystack</i> where it always matches (but it does not match after a newline <i>at the very end of Haystack</i>).</li><li>2) Dollar-sign (\$) matches before any newlines in <i>Haystack</i> (as well as at the very end where it always matches).</li></ol> <p>For example, the pattern "<b>m</b>)^abc\$" would not match the <i>Haystack</i>"xyz`r`nabc" unless the "m" option is present.</p> <p>The "D" option is ignored when "m" is present.</p> |
| <b>s</b> | DotAll. This causes a period (.) to match all characters including newlines (normally, it does not match newlines). However, when the newline character is at its default of CRLF (`r`n), two dots are required to match it (not one). Regardless of this option, a negative class such as                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | [^a] always matches newlines.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>x</b> | Ignores whitespace characters in the pattern except when escaped or inside a character class. The characters <code>\n</code> and <code>\t</code> are among those ignored because by the time they get to PCRE, they are already raw/literal whitespace characters (by contrast, <code>\n</code> and <code>\t</code> are not ignored because they are PCRE escape sequences). The <code>x</code> option also ignores characters between a non-escaped <code>#</code> outside a character class and the next newline character, inclusive. This makes it possible to include comments inside complicated patterns. However, this applies only to data characters; whitespace may never appear within special character sequences such as <code>(?()</code> , which begins a conditional subpattern. |
| <b>A</b> | Forces the pattern to be anchored; that is, it can match only at the start of <i>Haystack</i> . Under most conditions, this is equivalent to explicitly anchoring the pattern by means such as <code>^</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>D</b> | Forces dollar-sign (\$) to match at the very end of <i>Haystack</i> , even if <i>Haystack</i> 's last item is a newline. Without this option, \$ instead matches right before the final newline (if there is one). Note: This option is ignored when the "m" option is present.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>J</b> | Allows duplicate <a href="#">named subpatterns</a> . This can be useful for patterns in which only one of a collection of identically-named subpatterns can match. Note: If more than one instance of a particular name matches something, only the leftmost one is stored. Also, variable names are not case-sensitive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>U</b> | Ungreedy. Makes the quantifiers <code>*+?{ }</code> consume only those characters absolutely necessary to form a match, leaving the remaining ones available for the next part of the pattern. When the "U" option is not in effect, an individual quantifier can be made non-greedy by following it with a question mark. Conversely, when "U" is in effect, the question mark makes an individual quantifier greedy.                                                                                                                                                                                                                                                                                                                                                                            |
| <b>X</b> | PCRE_EXTRA. Enables PCRE features that are incompatible with Perl. Currently, the only such feature is that any backslash in a pattern that is followed by a letter that has no special meaning causes the match to fail and ErrorLevel to be set accordingly. This option helps reserve unused backslash sequences for future use. Without this option, a backslash followed by a letter with no special meaning is treated as a literal (e.g. <code>\g</code>                                                                                                                                                                                                                                                                                                                                   |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | and g are both recognized as a literal g). Regardless of this option, non-alphabetic backslash sequences that have no special meaning are always treated as literals (e.g. \ and / are both recognized as forward-slash).                                                                                                                                                                                                                                                                        |
| S  | Studies the pattern to try improve its performance. This is useful when a particular pattern (especially a complex one) will be executed many times. If PCRE finds a way to improve performance, that discovery is stored alongside the pattern in the cache for use by subsequent executions of the same pattern (subsequent uses of that pattern should also specify the S option because finding a match in the cache requires that the option letters exactly match, including their order). |
| C  | Enables the auto-callout mode. See <a href="#">Regular Expression Callouts</a> for more info.                                                                                                                                                                                                                                                                                                                                                                                                    |
|    | Enables recognition of additional newline markers. By default, only \r\n, \n and \r are recognized. With this option enabled, \v/VT/vertical tab/chr(0xB), \f/FF/formfeed/chr(0xC), NEL/next-line/chr(0x85), LS/line separator/chr(0x2028) and PS/paragraph separator/chr(0x2029) are also recognized.                                                                                                                                                                                           |
| \a | The \a, \n and \r options affect the behavior of <a href="#">anchors</a> (^ and \$) and the <a href="#">dot/period pattern</a> .<br><br>\a also puts (*BSR_UNICODE) into effect, which causes \R to match any kind of newline. By default, \R matches \n, \r and \r\n; this behaviour can be restored by combining options as follows: \a (*BSR_ANYCRLF)                                                                                                                                         |
| \n | Causes a solitary linefeed (\n) to be the only recognized newline marker (see above).                                                                                                                                                                                                                                                                                                                                                                                                            |
| \r | Causes a solitary carriage return (\r) to be the only recognized newline marker (see above).                                                                                                                                                                                                                                                                                                                                                                                                     |

Note: Spaces and tabs may optionally be used to separate each option from the next.

## Commonly Used Symbols and Syntax

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .         | <p>By default, a dot matches any single character which is not a newline character or part of a newline (<code>\r\n</code>) sequence, but this can be changed by using the <code>DotAll (s)</code>, <code>linefeed (f)</code>, <code>carriage return (r)</code>, or <code>\a</code> options. For example, <code>ab.</code> matches <code>abc</code> and <code>abz</code> and <code>ab</code>.</p>                                                                                                                                                                                                             |
| *         | <p>An asterisk matches zero or more of the preceding character, <code>class</code>, or <code>subpattern</code>. For example, <code>a*</code> matches <code>ab</code> and <code>aaab</code>. It also matches at the very beginning of any string that contains no "a" at all.</p> <p><b>Wildcard:</b> The dot-star pattern <code>.*</code> is one of the most permissive because it matches zero or more occurrences of <i>any</i> character (except newline: <code>\r</code> and <code>\n</code>). For example, <code>abc.*123</code> matches <code>abcAnything123</code> as well as <code>abc123</code>.</p> |
| ?         | <p>A question mark matches zero or one of the preceding character, <code>class</code>, or <code>subpattern</code>. Think of this as "the preceding item is optional". For example, <code>colou?r</code> matches both <code>color</code> and <code>colour</code> because the "u" is optional.</p>                                                                                                                                                                                                                                                                                                              |
| +         | <p>A plus sign matches one or more of the preceding character, <code>class</code>, or <code>subpattern</code>. For example <code>a+</code> matches <code>ab</code> and <code>aaab</code>. But unlike <code>a*</code> and <code>a?</code>, the pattern <code>a+</code> does not match at the beginning of strings that lack an "a" character.</p>                                                                                                                                                                                                                                                              |
| {min,max} | <p>Matches between <i>min</i> and <i>max</i> occurrences of the preceding character, <code>class</code>, or <code>subpattern</code>. For example, <code>a{1,2}</code> matches <code>ab</code> but only the first two a's in <code>aaab</code>.</p> <p>Also, <code>{3}</code> means exactly 3 occurrences, and <code>{3,}</code> means 3 or more occurrences. Note: The specified numbers must be less than 65536, and the first must be less than or equal to the second.</p>                                                                                                                                 |

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [...]  | <p><b>Classes of characters:</b> The square brackets enclose a list or range of characters (or both). For example, <code>[abc]</code> means "any single character that is either a, b or c". Using a dash in between creates a range; for example, <code>[a-z]</code> means "any single character that is between lowercase a and z (inclusive)". Lists and ranges may be combined; for example <code>[a-zA-Z0-9_]</code> means "any single character that is alphanumeric or underscore".</p> <p>A character class may be followed by <code>*</code>, <code>?</code>, <code>+</code>, or <code>{min,max}</code>. For example, <code>[0-9]+</code> matches one or more occurrence of any digit; thus it matches <code>xyz23</code> but not <code>abcxyz</code>.</p> <p>The following POSIX named sets are also supported via the form <code>[[:xxx:]]</code>, where <code>xxx</code> is one of the following words: <code>alnum</code>, <code>alpha</code>, <code>ascii</code> (0-127), <code>blank</code> (space or tab), <code>cntrl</code> (control character), <code>digit</code> (0-9), <code>xdigit</code> (hex digit), <code>print</code>, <code>graph</code> (print excluding space), <code>punct</code>, <code>lower</code>, <code>upper</code>, <code>space</code> (whitespace), <code>word</code> (same as <code>\w</code>).</p> <p>Within a character class, characters do not need to be escaped except when they have special meaning inside a class; e.g. <code>[^a]</code>, <code>[a\b]</code>, <code>[a\]]</code>, and <code>[\\a]</code>.</p> |
| [^...] | <p>Matches any single character that is <b>not</b> in the class. For example, <code>[^/]*</code> matches zero or more occurrences of any character that is <i>not</i> a forward-slash, such as <code>http://</code>. Similarly, <code>[^0-9xyz]</code> matches any single character that isn't a digit and isn't the letter <code>x</code>, <code>y</code>, or <code>z</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| \d     | <p>Matches any single digit (equivalent to the class <code>[0-9]</code>). Conversely, capital <code>\D</code> means "any <i>non</i>-digit". This and the other two below can also be used inside a <a href="#">class</a>; for example, <code>[\d.-]</code> means "any single digit, period, or minus sign".</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| \s     | <p>Matches any single whitespace character, mainly space, tab, and newline (<code>\r</code> and <code>\n</code>). Conversely, capital <code>\S</code> means "any <i>non</i>-whitespace character".</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \w      | Matches any single "word" character, namely alphanumeric or underscore. This is equivalent to <code>[a-zA-Z0-9_]</code> . Conversely, capital <code>\W</code> means "any <i>non</i> -word character".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| ^<br>\$ | <p>Circumflex (^) and dollar sign (\$) are called <i>anchors</i> because they don't consume any characters; instead, they tie the pattern to the beginning or end of the string being searched.</p> <p><code>^</code> may appear at the beginning of a pattern to require the match to occur at the very beginning of a line. For example, <code>^abc</code> matches <code>abc123</code> but not <code>123abc</code>.</p> <p><code>\$</code> may appear at the end of a pattern to require the match to occur at the very end of a line. For example, <code>abc\$</code> matches <code>123abc</code> but not <code>abc123</code>.</p> <p>The two anchors may be combined. For example, <code>^abc\$</code> matches only <code>abc</code> (i.e. there must be no other characters before or after it).</p> <p>If the text being searched contains multiple lines, the anchors can be made to apply to each line rather than the text as a whole by means of the "m" option. For example, <code>m)^abc\$</code> matches <code>123\r\nabc\r\n789</code>. But without the "m" option, it wouldn't match.</p> |
| \b      | \b means "word boundary", which is like an anchor because it doesn't consume any characters. It requires the current character's <i>status as a word character</i> ( <code>\w</code> ) to be the opposite of the previous character's. It is typically used to avoid accidentally matching a word that appears inside some other word. For example, <code>\bcat\b</code> doesn't match <code>catfish</code> , but it matches <code>cat</code> regardless of what punctuation and whitespace surrounds it. Capital <code>\B</code> is the opposite: it requires that the current character <i>not</i> be at a word boundary.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         | The vertical bar separates two or more alternatives. A match occurs if <i>any</i> of the alternatives is satisfied. For example, <code>gray grey</code> matches both <code>gray</code> and <code>grey</code> . Similarly, the pattern                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                     | <p><code>gr(a e)y</code> does the same thing with the help of the parentheses described below.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <p>(...)</p>                                        | <p>Items enclosed in parentheses are most commonly used to:</p> <ul style="list-style-type: none"> <li>• Determine the order of evaluation. For example, <code>(Sun Mon Tues Wednes Thurs Fri Sat)day</code> matches the name of any day.</li> <li>• Apply <code>*</code>, <code>?</code>, <code>+</code>, or <code>{min,max}</code> to a <i>series</i> of characters rather than just one. For example, <code>(abc)+</code> matches one or more occurrences of the string "abc"; thus it matches <code>abcabc123</code> but not <code>ab123</code> or <code>bc123</code>.</li> <li>• Capture a subpattern such as the dot-star in <code>abc(.*)xyz</code>. For example, <code>RegexMatch()</code> stores the substring that matches each subpattern in its <i>output array</i>. Similarly, <code>RegexReplace()</code> allows the substring that matches each subpattern to be reinserted into the result via <i>backreferences</i> like <code>\$1</code>. To use the parentheses without the side-effect of capturing a subpattern, specify <code>?:</code> as the first two characters inside the parentheses; for example: <code>?:*</code></li> <li>• Change <i>options</i> on-the-fly. For example, <code>(?im)</code> turns on the case-insensitive and multiline options for the remainder of the pattern (or subpattern if it occurs inside a subpattern). Conversely, <code>(?-im)</code> would turn them both off. All options are supported except <code>DPS`r`n`a</code>.</li> </ul> |
| <p><code>\t</code><br/><code>\r</code><br/>etc.</p> | <p>These escape sequences stand for special characters. The most common ones are <code>\t</code> (tab), <code>\r</code> (carriage return), and <code>\n</code> (linefeed). In AutoHotkey, an accent (<code>^</code>) may optionally be used in place of the backslash in these cases. Escape sequences in the form <code>\xhh</code> are also supported, in which <i>hh</i> is the hex code of any ANSI character between 00 and FF.</p> <p><code>\R</code> matches <code>\r\n</code>, <code>\n</code> and <code>\r</code> (however, <code>\R</code> inside a <i>character class</i> is merely the letter "R").</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

|                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><code>\p{xx}</code><br/> <code>\P{xx}</code><br/> <code>\X</code></p> | <p>Unicode character properties. <code>\p{xx}</code> matches a character with the xx property while <code>\P{xx}</code> matches any character <i>without</i> the xx property. For example, <code>\pL</code> matches any letter and <code>\p{Lu}</code> matches any upper-case letter. <code>\X</code> matches any number of characters that form an extended Unicode sequence.</p> <p>For a full list of supported property names and other details, search for "<code>\p{xx}</code>" at <a href="http://www.pcre.org/pcre.txt">www.pcre.org/pcre.txt</a>.</p> |
| <p><code>(*UCP)</code></p>                                               | <p>For performance, <code>\d</code>, <code>\D</code>, <code>\s</code>, <code>\S</code>, <code>\w</code>, <code>\W</code>, <code>\b</code> and <code>\B</code> recognize only ASCII characters by default. If the pattern begins with <code>(*UCP)</code>, Unicode properties will be used to determine which characters match. For example, <code>\w</code> becomes equivalent to <code>\p{L}\p{N}_</code> and <code>\d</code> becomes equivalent to <code>\p{Nd}</code>.</p>                                                                                  |

**Greed:** By default, `*`, `?`, `+`, and `{min,max}` are greedy because they consume all characters up through the *last* possible one that still satisfies the entire pattern. To instead have them stop at the *first* possible character, follow them with a question mark. For example, the pattern `<.+>` (which lacks a question mark) means: "search for a `<`, followed by one or more of any character, followed by a `>`". To stop this pattern from matching the *entire* string `<em>text</em>`, append a question mark to the plus sign: `<.+?>`. This causes the match to stop at the first `'>`' and thus it matches only the first tag `<em>`.

**Look-ahead and look-behind assertions:** The groups `(?=...)`, `(?!...)`, `(?<=...)`, and `(?<!...)` are called *assertions* because they demand a condition to be met but don't consume any characters. For example, `abc(?=.*xyz)` is a look-ahead assertion that requires the string xyz to exist somewhere to the right of the string abc (if it doesn't, the entire pattern is not considered a match). `(?=...)` is called a

*positive* look-ahead because it requires that the specified pattern exist. Conversely, `(?!...)` is a *negative* look-ahead because it requires that the specified pattern *not* exist. Similarly, `(?<=...)` and `(?<!...)` are positive and negative look-behinds (respectively) because they look to the *left* of the current position rather than the right. Look-behinds are more limited than look-aheads because they do not support quantifiers of varying size such as `*`, `?`, and `+`. The escape sequence `\K` is similar to a look-behind assertion because it causes any previously-matched characters to be omitted from the final matched string. For example, `foo\Kbar` matches "foobar" but reports that it has matched "bar".

**Related:** Regular expressions are supported by [RegexMatch\(\)](#), [RegexReplace\(\)](#), and [SetTitleMatchMode](#).

**Final note:** Although this page touches upon most of the commonly-used RegEx features, there are quite a few other features you may want to explore such as conditional subpatterns. The complete PCRE manual is at [www.pcre.org/pcre.txt](http://www.pcre.org/pcre.txt)

# Acknowledgements

In addition to the AutoIt authors already [mentioned](#):

**Robert Yaklin:** A lot of tireless testing to isolate bugs, a great first draft of the installer, as well as great suggestions for how commands **ought** to work :)

**Jason Payam Ahdoot:** For suggesting and describing floating point support.

**Jay D. Novak:** For discovering many Win9x problems with the Send command, Capslock, and hotkey modifiers; and for generously sharing his wealth of code and wisdom for hotkeys, hot-strings, hook usage, and typing acceleration.

**Rajat:** For creating stylish replacements for the original AHK icons; a great product logo; making the syntax customizations for TextPad; discovering some bugs with the registry commands and AutoIt v2 compatibility; making SmartGUI Creator; and many other things.

**Thaddeus Beck (Beardboy):** For NT4 testing to fix GetKeyState and the Send command; for a lot of help on the forum; and for many suggestions and bug reports.

**Gregory F. Hogg of Hogg's Software:** For writing the source code for multi-monitor support in the [SysGet](#) command.

**Aurelian Maga:** For writing the source code for [ImageSearch](#) and the faster [PixelSearch](#).

**Joost Mulders:** Whose source code provided the foundation for [expressions](#).

**Laszlo Hars:** For advice about data structures and algorithms, which helped greatly speed up arrays and dynamic variables.

**Marcus Sonntag (Ultra):** For the research, design, coding, and testing of [DllCall](#).

**Gena Shimanovich:** For debugging and coding assistance; and for some advanced prototype scripts upon which future features may be based.

**Eric Morin (numEric):** For advice on mathematics; for steadfast debugging of areas like [OnMessage](#), floating point computations, and mouse movement; and for improvements to overall quality.

**Philip Hazel:** For [Perl-Compatible Regular Expressions \(PCRE\)](#).

**Titan/polyethene:** For providing community hosting on [autohotkey.net](#), creating many useful scripts and libraries, creating the JavaScript to colorize code-comments in the forum, and many other things.

**Philippe Lhoste (PhiLho):** For tireless moderation and support in the forum, RegEx advice and testing, syntax and design ideas, and many other things.

**John Biederman:** For greatly improving the presentation and ergonomics of the documentation.

**Jonathan Rennison (JGR):** For developing [RegisterCallback](#), and for beneficial suggestions.

**Steve Gray (Lexikos):** For developing dynamic function calling and other new functionality; analyzing and fixing bugs; and valuable support and advice for hundreds of individual visitors at the forum.

AutoHotkey\_L:

**jackeiku:** For developing Unicode support and other new functionality.

**fincs:** For developing native 64-bit support and try/catch/throw, writing a replacement for the old compiler, analyzing and fixing bugs.

**Sean:** For developing built-in COM functionality and providing valuable insight into COM.

**TheGood:** For merging the documentation and adapting the installer script.

**ac:** For developing #Warn.

**Russell Davis:** For developing A\_PriorKey, ahk\_path (the basis of ahk\_exe in WinTitle parameters), #InputLevel and SendLevel.

**Christian Sander:** For developing support for SysLink controls.

*And to everyone else who's contributed or sent in bug reports or suggestions:  
Thanks!*

# Creating a Keyboard Macro or Mouse Macro

A macro is a series of scripted actions that is "played" upon demand. The most common activity of a macro is to send [simulated keystrokes](#) and [mouse clicks](#) to one or more windows. Such windows respond to each keystroke and mouse click as though you had performed it manually, which allows repetitive tasks to be automated with high speed and reliability.

Although macros can be written by hand, you might find it easier to write long ones with the aid of a macro recorder such as the [Recorder script](#) written by Titan/polyethene. It watches what you type and where you click, and keeps track of which window is [active](#). It transcribes these actions into a working macro that can later be "played back" at a faster speed.

One of the most convenient ways to play back a macro is to assign it to a [hotkey](#) or [hotstring](#). For example, the following hotkey would create an empty e-mail message and prepare it for a certain type recipient, allowing you to personalize it prior to sending:

```
^!s:: ; Control+Alt+S hotkey.
if !WinExist("Inbox - Microsoft Outlook")
 return ; Outlook isn't open to the right
section, so do nothing.
WinActivate ; Activate the window found by the
above command.
Send ^n ; Create new/blank e-mail via
Control+N.
```

```
WinWaitActive Untitled Message
Send {Tab 2}Product Recall for ACME Rocket
Skates ; Set the subject line.
Send {Tab}Dear Sir or Madam,{Enter 2}We have
recently discovered a minor defect ... ; etc.
return ; This line serves to finish the
hotkey.
```

Hotkey macros like the above are especially useful for tasks you perform several times per day. By contrast, macros used less often can each be kept in a separate script accessible by means of a shortcut in the Start Menu or on the desktop.

To start creating your own macros and hotkeys right away, please read the [Quick-start Tutorial](#).

-- Home --

# Overriding or Disabling Hotkeys

You can disable all built-in Windows hotkeys except Win+L and Win+U by making the following change to the registry (this should work on all OSes but a reboot is probably required):

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
NoWinKeys REG_DWORD 0x00000001 (1)
```

But read on if you want to do more than just disable them all.

Hotkeys owned by another application can be overridden or disabled simply by assigning them to an action in the script. The most common use for this feature is to change the hotkeys that are built into Windows itself. For example, if you wish Win+E (the shortcut key that launches Windows Explorer) to perform some other action, use this:

```
#e::
MsgBox This hotkey is now owned by the script.
return
```

In the next example, the Win+R hotkey, which is used to open the RUN window, is completely disabled:

```
#r::return
```

Similarly, to disable both Windows keys, use this:

```
Lwin::return
Rwin::return
```

To disable or change an application's non-global hotkey (that is, a shortcut key that only works when that application is the active window), consider the following example which disables Control+P (Print) only for Notepad, leaving the key in effect for all other types of windows:

```
$^p::
if WinActive("ahk_class Notepad")
 return ; i.e. do nothing, which causes
Control-P to do nothing in Notepad.
Send ^p
return
```

In the above example, the \$ prefix is needed so that the hotkey can "send itself" without activating itself (which would otherwise trigger a warning dialog about an infinite loop). See also: [context-sensitive hotkeys](#).

You can try out any of the above examples by copying them into a new text file such as "Override.ahk", then launching the file.

# Threads

The *current thread* is defined as the flow of execution invoked by the most recent event; examples include [hotkeys](#), [SetTimer](#) subroutines, [custom menu items](#), and [GUI events](#). The *current thread* can be executing commands within its own subroutine or within other subroutines called by that subroutine.

Although AutoHotkey doesn't actually use multiple threads, it simulates some of that behavior: If a second thread is started -- such as by pressing another hotkey while the previous is still running -- the *current thread* will be interrupted (temporarily halted) to allow the new thread to become *current*. If a third thread is started while the second is still running, both the second and first will be in a dormant state, and so on.

When the *current thread* finishes, the one most recently interrupted will be resumed, and so on, until all the threads finally finish. When resumed, a thread's settings for things such as [ErrorLevel](#) and [SendMode](#) are automatically restored to what they were just prior to its interruption; in other words, a thread will experience no side-effects from having been interrupted (except for a possible change in the [active window](#)).

Note: The [KeyHistory](#) command/menu-item shows how many threads are in an interrupted state and the [ListHotkeys](#) command/menu-item shows which hotkeys have threads.

A single script can have multiple simultaneous [MsgBox](#), [InputBox](#), [FileSelect](#), and [DirSelect](#) dialogs. This is achieved by launching a new thread (via [hotkey](#),

timed subroutine, custom menu item, etc.) while a prior thread already has a dialog displayed.

By default, a given hotkey or hotstring subroutine cannot be run a second time if it is already running. Use #MaxThreadsPerHotkey to change this behavior.

## Thread Priority

Any thread ([hotkey](#), [timed subroutine](#), [custom menu item](#), etc.) with a priority lower than that of the *current thread* cannot interrupt it. During that time, such timers will not run, and any attempt by the user to create a thread (such as by pressing a [hotkey](#) or [GUI button](#)) will have no effect, nor will it be buffered. Because of this, it is usually best to design high priority threads to finish quickly, or use [Critical](#) instead of making them high priority.

The default priority is 0. All threads use the default priority unless changed by one of the following methods:

- 1) A timed subroutine is given a specific priority via [SetTimer](#).
- 2) A hotkey is given a specific priority via the [Hotkey](#) command.
- 3) A [hotstring](#) is given a specific priority when it is defined, or via the [#Hotstring](#) directive.
- 4) A custom menu item is given a specific priority via the [Menu](#) command.
- 5) The *current thread* sets its own priority via the [Thread](#) command.

The [OnExit](#) callback function (if any) will always run when called for, regardless of the *current thread's* priority.

# Escape Sequences

The escape character ``` (back-tick or grave accent) is used to indicate that the character immediately following it should be interpreted differently than it normally would.

| Type This       | To Get This                                                                                                                                                                                                                                                                                          |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>``</code> | <code>`</code> (literal accent; i.e. two consecutive escape characters result in a single literal character)                                                                                                                                                                                         |
| <code>`;</code> | <code>;</code> (literal semicolon). <b>Note:</b> This is necessary <u>only</u> if a semicolon has a space or tab to its left. If it does not, it will be recognized correctly without being escaped.                                                                                                 |
| <code>`n</code> | newline (linefeed/LF)                                                                                                                                                                                                                                                                                |
| <code>`r</code> | carriage return (CR)                                                                                                                                                                                                                                                                                 |
| <code>`b</code> | backspace                                                                                                                                                                                                                                                                                            |
| <code>`t</code> | tab (the more typical horizontal variety)                                                                                                                                                                                                                                                            |
| <code>`s</code> | space                                                                                                                                                                                                                                                                                                |
| <code>`v</code> | vertical tab -- corresponds to Ascii value 11. It can also be manifest in some applications by typing Control+K.                                                                                                                                                                                     |
| <code>`a</code> | alert (bell) -- corresponds to Ascii value 7. It can also be manifest in some applications by typing Control+G.                                                                                                                                                                                      |
| <code>`f</code> | formfeed -- corresponds to Ascii value 12. It can also be manifest in some applications by typing Control+L.                                                                                                                                                                                         |
| Send            | When the <a href="#">Send command</a> or <a href="#">Hotstrings</a> are used in their default (non-raw) mode, characters such as <code>{ } ^ ! + #</code> have special meaning. Therefore, to use them literally in these cases, enclose them in braces. For example: <code>Send {^} ! {}{}</code> . |
|                 | Single-quote marks (') and double-quote marks (") function                                                                                                                                                                                                                                           |

``" or `"`

identically, except that a string enclosed in single-quote marks can contain literal double-quote marks and vice versa. Therefore, to include an actual quote mark inside a literal string, escape the quote mark or enclose the string in the opposite type of quote mark. For example: `Var := "The color `red` was found."` or `Var := 'The color "red" was found.'`.

## Example

```
MsgBox "Line 1`nLine 2"
```

# Using Keyboard Numpad as a Mouse -- by deguix

This script makes mousing with your keyboard almost as easy as using a real mouse (maybe even easier for some tasks). It supports up to five mouse buttons and the turning of the mouse wheel. It also features customizable movement speed, acceleration, and "axis inversion".

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
/*

-----0
|Using Keyboard Numpad as a Mouse
|
|
|-----
|-----)
| By deguix / A Script file for
AutoHotkey
This script is an example of use of
AutoHotkey. It uses
the remapping of numpad keys of a keyboard to
transform it
into a mouse. Some features are the acceleration
which
enables you to increase the mouse movement when
holding
a key for a long time, and the rotation which
```

makes the  
| numpad mouse to "turn". I.e. NumPadDown as  
NumPadUp  
| and vice-versa. See the list of keys used below:

| Keys | Description |
|------|-------------|
|------|-------------|

|                        |                              |
|------------------------|------------------------------|
| ScrollLock (toggle on) | Activates numpad mouse mode. |
|------------------------|------------------------------|

|         |                          |
|---------|--------------------------|
| NumPad0 | Left mouse button click. |
|---------|--------------------------|

|         |                            |
|---------|----------------------------|
| NumPad5 | Middle mouse button click. |
|---------|----------------------------|

|           |                           |
|-----------|---------------------------|
| NumPadDot | Right mouse button click. |
|-----------|---------------------------|

|                      |                           |
|----------------------|---------------------------|
| NumPadDiv/NumPadMult | X1/X2 mouse button click. |
|----------------------|---------------------------|

|                     |                                |
|---------------------|--------------------------------|
| NumPadSub/NumPadAdd | Moves up/down the mouse wheel. |
|---------------------|--------------------------------|

|                       |                                |
|-----------------------|--------------------------------|
| NumLock (toggled off) | Activates mouse movement mode. |
|-----------------------|--------------------------------|

|                      |                 |
|----------------------|-----------------|
| NumPadEnd/Down/PgDn/ | Mouse movement. |
|----------------------|-----------------|



```
to |
| | right in degrees. (i.e.
180°= |
| | = inversed controls).
|

* = These options are affected by the mouse
wheel speed
adjusted on Control Panel. If you don't have a
mouse with
wheel, the default is 3 +/- lines per option
button press. |
0-----|
-----0
*/
```

```
;START OF CONFIG SECTION
```

```
#SingleInstance
#MaxHotkeysPerInterval 500
```

```
; Using the keyboard hook to implement the Numpad
hotkeys prevents
; them from interfering with the generation of
ANSI characters such
; as à. This is because AutoHotkey generates such
characters
; by holding down ALT and sending a series of
Numpad keystrokes.
; Hook hotkeys are smart enough to ignore such
keystrokes.
```

```
#UseHook
```

```
MouseSpeed := 1
MouseAccelerationSpeed := 1
MouseMaxSpeed := 5
```

```
;Mouse wheel speed is also set on Control Panel.
As that
;will affect the normal mouse behavior, the real
speed of
;these three below are times the normal mouse
wheel speed.
```

```
MouseWheelSpeed := 1
MouseWheelAccelerationSpeed := 1
MouseWheelMaxSpeed := 5
```

```
MouseRotationAngle := 0
```

```
;END OF CONFIG SECTION
```

```
;This is needed or key presses would faulty send
their natural
;actions. Like NumPadDiv would send sometimes "/"
to the
;screen.
```

```
#InstallKeybdHook
```

```
Temp := 0
Temp2 := 0
```

```
MouseRotationAnglePart := MouseRotationAngle
;Divide by 45° because MouseMove only supports
whole numbers,
;and changing the mouse rotation to a number
lesser than 45°
;could make strange movements.
```

```
;
;For example: 22.5° when pressing NumPadUp:
; First it would move upwards until the speed
; to the side reaches 1.
```

```
MouseRotationAnglePart /= 45
```



```

Hotkey, Numpad6, ButtonRotationAngleUp
Hotkey, Numpad4, ButtonRotationAngleDown

Hotkey, !Numpad8, ButtonWheelSpeedUp
Hotkey, !Numpad2, ButtonWheelSpeedDown
Hotkey, !Numpad7, ButtonWheelAccelerationSpeedUp
Hotkey, !Numpad1, ButtonWheelAccelerationSpeedDown
Hotkey, !Numpad9, ButtonWheelMaxSpeedUp
Hotkey, !Numpad3, ButtonWheelMaxSpeedDown

Gosub, ~ScrollLock ; Initialize based on current
ScrollLock state.
return

;Key activation support

~ScrollLock::
; Wait for it to be released because otherwise the
hook state gets reset
; while the key is down, which causes the up-event
to get suppressed,
; which in turn prevents toggling of the
ScrollLock state/light:
KeyWait, ScrollLock
GetKeyState, ScrollLockState, ScrollLock, T
If ScrollLockState
{
 Hotkey, *NumPad0, on
 Hotkey, *NumpadIns, on
 Hotkey, *NumPad5, on
 Hotkey, *NumPadDot, on
 Hotkey, *NumPadDel, on
 Hotkey, *NumPadDiv, on
 Hotkey, *NumPadMult, on

 Hotkey, *NumpadSub, on
 Hotkey, *NumpadAdd, on

```



```
Hotkey, *NumpadSub, off
Hotkey, *NumpadAdd, off

Hotkey, *NumPadUp, off
Hotkey, *NumPadDown, off
Hotkey, *NumPadLeft, off
Hotkey, *NumPadRight, off
Hotkey, *NumPadHome, off
Hotkey, *NumPadEnd, off
Hotkey, *NumPadPgUp, off
Hotkey, *NumPadPgDn, off

Hotkey, Numpad8, off
Hotkey, Numpad2, off
Hotkey, Numpad7, off
Hotkey, Numpad1, off
Hotkey, Numpad9, off
Hotkey, Numpad3, off

Hotkey, Numpad6, off
Hotkey, Numpad4, off

Hotkey, !Numpad8, off
Hotkey, !Numpad2, off
Hotkey, !Numpad7, off
Hotkey, !Numpad1, off
Hotkey, !Numpad9, off
Hotkey, !Numpad3, off
```

```
}
return
```

**;Mouse click support**

```
ButtonLeftClick:
GetKeyState, already_down_state, LButton
If already_down_state
 return
```



```

Button2 := "NumPadDel"
ButtonClick := "Right"
Goto ButtonClickStart

ButtonX1Click:
GetKeyState, already_down_state, XButton1
If already_down_state
 return
Button2 := "NumPadDiv"
ButtonClick := "X1"
Goto ButtonClickStart

ButtonX2Click:
GetKeyState, already_down_state, XButton2
If already_down_state
 return
Button2 := "NumPadMult"
ButtonClick := "X2"
Goto ButtonClickStart

ButtonClickStart:
MouseClicked, %ButtonClick%, , , 1, 0, D
SetTimer, ButtonClickEnd, 10
return

ButtonClickEnd:
GetKeyState, kclickstate, %Button2%, P
if kclickstate
 return

SetTimer, , off
MouseClicked, %ButtonClick%, , , 1, 0, U
return

;Mouse movement support

ButtonSpeedUp:

```





```

ButtonUpLeft:
ButtonUpRight:
ButtonDownLeft:
ButtonDownRight:
If Button <> 0
{
 if !InStr(A_ThisHotkey, Button)
 {
 MouseCurrentAccelerationSpeed := 0
 MouseCurrentSpeed := MouseSpeed
 }
}
StrReplace, Button, %A_ThisHotkey%, *

ButtonAccelerationStart:
If MouseAccelerationSpeed >= 1
{
 If MouseMaxSpeed > MouseCurrentSpeed
 {
 Temp := 0.001
 Temp *= MouseAccelerationSpeed
 MouseCurrentAccelerationSpeed += Temp
 MouseCurrentSpeed +=
MouseCurrentAccelerationSpeed
 }
}

```

**;MouseRotationAngle conversion to speed of button direction**

```

{
 MouseCurrentSpeedToDirection :=
MouseRotationAngle
 MouseCurrentSpeedToDirection /= 90.0
 Temp := MouseCurrentSpeedToDirection

 if Temp >= 0
 {

```



```

 Goto
EndMouseCurrentSpeedToDirectionCalculation
 }
}
EndMouseCurrentSpeedToDirectionCalculation:
;MouseRotationAngle conversion to speed of 90
degrees to right
{
 MouseCurrentSpeedToSide := MouseRotationAngle
 MouseCurrentSpeedToSide /= 90.0
 Temp := Mod(MouseCurrentSpeedToSide, 4)

 if Temp >= 0
 {
 if Temp < 1
 {
 MouseCurrentSpeedToSide := 0
 MouseCurrentSpeedToSide += Temp
 Goto
EndMouseCurrentSpeedToSideCalculation
 }
 }
 if Temp >= 1
 {
 if Temp < 2
 {
 MouseCurrentSpeedToSide := 1
 Temp -= 1
 MouseCurrentSpeedToSide -= Temp
 Goto
EndMouseCurrentSpeedToSideCalculation
 }
 }
 if Temp >= 2
 {

```







```
 Temp2 += MouseCurrentSpeedToSide
 MouseMove, %Temp%, %Temp2%, 0, R
}
```

```
SetTimer, ButtonAccelerationEnd, 10
return
```

```
ButtonAccelerationEnd:
GetKeyState, kstate, %Button%, P
if kstate
 Goto ButtonAccelerationStart
```

```
SetTimer,, off
MouseCurrentAccelerationSpeed := 0
MouseCurrentSpeed := MouseSpeed
Button := 0
return
```

### **;Mouse wheel movement support**

```
ButtonWheelSpeedUp:
MouseWheelSpeed++
RegRead, MouseWheelSpeedMultiplier, HKCU\Control
Panel\Desktop, WheelScrollLines
If MouseWheelSpeedMultiplier <= 0
 MouseWheelSpeedMultiplier := 1
MouseWheelSpeedReal := MouseWheelSpeed
MouseWheelSpeedReal *= MouseWheelSpeedMultiplier
ToolTip, Mouse wheel speed: %MouseWheelSpeedReal%
lines
SetTimer, RemoveToolTip, 1000
return
```

```
ButtonWheelSpeedDown:
RegRead, MouseWheelSpeedMultiplier, HKCU\Control
Panel\Desktop, WheelScrollLines
If MouseWheelSpeedMultiplier <= 0
 MouseWheelSpeedMultiplier := 1
```









# Remapping a Joystick to Keyboard or Mouse

## Table of Contents

- [Important Notes](#)
- [Making a Joystick Button Send Keystrokes or Mouse Clicks](#)
- [Making a Joystick Axis or POV Hat Send Keystrokes or Mouse Clicks](#)
- [Remarks](#)

## Important Notes

- Although a joystick button or axis can be remapped to become a key or mouse button, it cannot be remapped to some other joystick button or axis. That would be possible only with the help of a joystick emulator such as [PPJoy](#).
- AutoHotkey identifies each button on a joystick with a unique number between 1 and 32. To determine these numbers, use the [joystick test script](#).

## Making a Joystick Button Send Keystrokes or Mouse Clicks

Below are three approaches, starting at the simplest and ending with the most complex. The most complex method works in the broadest variety of circumstances (such as games that require a key or mouse button to be held down).

**Method #1:** This method sends simple keystrokes and mouse clicks. For example:

```
Joy1: Send {Left} ; Have button #1 send a left-arrow keystroke.
Joy2: Click ; Have button #2 send a click of left mouse button.
Joy3::Send a{Esc}{Space}{Enter} ; Have button #3 send the letter "a" followed by Escape, Space, and Enter.
Joy4::Send Sincerely,{Enter}John Smith ; Have button #4 send a two-line signature.
```

To have a button perform more than one command, put the first command *beneath* the button name and make the last command a `return`. For example:

```
Joy5::
Run Notepad
WinWait Untitled - Notepad
WinActivate
Send This is the text that will appear in
Notepad.{Enter}
```

```
return
```

See the [Key List](#) for the complete list of keys and mouse/joystick buttons.

**Method #2:** This method is necessary in cases where a key or mouse button must be held down for the entire time that you're holding down a joystick button. The following example makes the joystick's second button become the left-arrow key:

```
Joy2::
Send {Left down} ; Hold down the left-arrow
key.
Keywait Joy2 ; Wait for the user to release
the joystick button.
Send {Left up} ; Release the left-arrow key.
return
```

**Method #3:** This method is necessary in cases where you have more than one joystick hotkey of the type described in Method #2, and you sometimes press and release such hotkeys simultaneously. The following example makes the joystick's third button become the left mouse button:

```
Joy3::
Send {LButton down} ; Hold down the left
mouse button.
SetTimer, WaitForButtonUp3, 10
return

WaitForButtonUp3:
```

```

if GetKeyState("Joy3") ; The button is still,
down, so keep waiting.
 return
; Otherwise, the button has been released.
Send {LButton up} ; Release the left mouse
button.
SetTimer,, off
return

```

**Auto-repeating a keystroke:** Some programs or games might require a key to be sent repeatedly (as though you are holding it down on the keyboard). The following example achieves this by sending spacebar keystrokes repeatedly while you hold down the joystick's second button:

```

Joy2::
Send {Space down} ; Press the spacebar down.
SetTimer, WaitForJoy2, 30 ; Reduce the number
30 to 20 or 10 to send keys faster. Increase it
to send slower.
return

WaitForJoy2:
if not GetKeyState("Joy2") ; The button has
been released.
{
 Send {Space up} ; Release the spacebar.
 SetTimer,, off ; Stop monitoring the
button.
 return
}
; Since above didn't "return", the button is
still being held down.

```

```
Send {Space down} ; Send another Spacebar
keystroke.
return
```

**Context-sensitive Joystick Buttons:** The directives [#IfWinActive/Exist](#) can be used to make selected joystick buttons perform a different action (or none at all) depending on the type of window that is active or exists.

**Using a Joystick as a Mouse:** The [Joystick-To-Mouse](#) script converts a joystick into a mouse by remapping its buttons and axis control.

## Making a Joystick Axis or POV Hat Send Keystrokes or Mouse Clicks

To have a script respond to movement of a joystick's axis or POV hat, use `SetTimer` and `GetKeyState`. The following example makes the joystick's X and Y axes behave like the arrow key cluster on a keyboard (left, right, up, and down):

```
SetTimer, WatchAxis, 5
return

WatchAxis:
GetKeyState, JoyX, JoyX ; Get position of X
axis.
GetKeyState, JoyY, JoyY ; Get position of Y
axis.
KeyToHoldDownPrev := KeyToHoldDown ; Prev now
holds the key that was down before (if any).

if JoyX > 70
 KeyToHoldDown := "Right"
else if JoyX < 30
 KeyToHoldDown := "Left"
else if JoyY > 70
 KeyToHoldDown := "Down"
else if JoyY < 30
 KeyToHoldDown := "Up"
else
 KeyToHoldDown := ""

if KeyToHoldDown = KeyToHoldDownPrev ; The
correct key is already down (or no key is
needed).
 return ; Do nothing.
```

```

; Otherwise, release the previous key and press
down the new key:
SetKeyDelay -1 ; Avoid delays between
keystrokes.
if KeyToHoldDownPrev ; There is a previous
key to release.
 Send, {%KeyToHoldDownPrev% up} ; Release
it.
if KeyToHoldDown ; There is a key to press
down.
 Send, {%KeyToHoldDown% down} ; Press it
down.
return

```

The following example makes the joystick's POV hat behave like the arrow key cluster on a keyboard; that is, the POV hat will send arrow keystrokes (left, right, up, and down):

```

SetTimer, WatchPOV, 5
return

WatchPOV:
GetKeyState, POV, JoyPOV ; Get position of the
POV control.
KeyToHoldDownPrev := KeyToHoldDown ; Prev now
holds the key that was down before (if any).

; Some joysticks might have a smooth/continuous
POV rather than one in fixed increments.
; To support them all, use a range:
if POV < 0 ; No angle to report

```

```

 KeyToHoldDown := ""
else if POV > 31500 ; 315 to 360
degrees: Forward
 KeyToHoldDown := "Up"
else if POV >= 0 and POV <= 4500 ; 0 to 45
degrees: Forward
 KeyToHoldDown := "Up"
else if POV >= 4501 and POV <= 13500 ; 45 to
135 degrees: Right
 KeyToHoldDown := "Right"
else if POV >= 13501 and POV <= 22500 ; 135 to
225 degrees: Down
 KeyToHoldDown := "Down"
else ; 225 to
315 degrees: Left
 KeyToHoldDown := "Left"

if KeyToHoldDown = KeyToHoldDownPrev ; The
correct key is already down (or no key is
needed).
 return ; Do nothing.

; Otherwise, release the previous key and press
down the new key:
SetKeyDelay -1 ; Avoid delays between
keystrokes.
if KeyToHoldDownPrev ; There is a previous
key to release.
 Send, {%KeyToHoldDownPrev% up} ; Release
it.
if KeyToHoldDown ; There is a key to press
down.
 Send, {%KeyToHoldDown% down} ; Press it
down.
return

```

**Auto-repeating a keystroke:** Both examples above can be modified to send the key repeatedly rather than merely holding it down (that is, they can mimic physically holding down a key on the keyboard). To do this, replace the following line:

```
return ; Do nothing.
```

WITH:

```
{
 if KeyToHoldDown
 Send, {%KeyToHoldDown% down} ; Auto-
repeat the keystroke.
 return
}
```

## Remarks

A joystick other than first may be used by preceding the button or axis name with the number of the joystick. For example, `2Joy1` would be the second joystick's first button.

To find other useful joystick scripts, visit the [AutoHotkey forum](#). A keyword search such as *Joystick and GetKeyState and Send* is likely to produce topics of interest.

## Related Topics

[Joystick-To-Mouse script \(using a joystick as a mouse\)](#)

[List of joystick buttons, axes, and controls](#)

[GetKeyState](#)

[Remapping the keyboard and mouse](#)

# Joystick Test Script

This script helps determine the button numbers and other attributes of your joystick. It might also reveal if your joystick is in need of calibration; that is, whether the range of motion of each of its axes is from 0 to 100 percent as it should be. If calibration is needed, use the operating system's control panel or the software that came with your joystick.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; July 16, 2016: Revised code for AHK v2
compatibility
; July 6, 2005 : Added auto-detection of joystick
number.
; May 8, 2005 : Fixed: JoyAxes is no longer
queried as a means of
; detecting whether the joystick is connected.
Some joysticks are
; gamepads and don't have even a single axis.

; If you want to unconditionally use a specific
joystick number, change
; the following value from 0 to the number of the
joystick (1-16).
; A value of 0 causes the joystick number to be
auto-detected:
JoystickNumber := 0

; END OF CONFIG SECTION. Do not make changes below
this point unless
; you wish to alter the basic functionality of the
script.
```

```

; Auto-detect the joystick number if called for:
if JoystickNumber <= 0
{
 Loop 16 ; Query each joystick number to find
out which ones exist.
 {
 GetKeyState, JoyName, %A_Index%JoyName
 if JoyName <> ""
 {
 JoystickNumber := A_Index
 break
 }
 }
 if JoystickNumber <= 0
 {
 MsgBox The system does not appear to have
any joysticks.
 ExitApp
 }
}

#SingleInstance
GetKeyState, joy_buttons,
%JoystickNumber%JoyButtons
GetKeyState, joy_name, %JoystickNumber%JoyName
GetKeyState, joy_info, %JoystickNumber%JoyInfo
Loop
{
 buttons_down := ""
 Loop, joy_buttons
 {
 GetKeyState, joy%a_index%,
%JoystickNumber%joy%a_index%
 if joy%a_index%
 buttons_down .= " " a_index
 }
 GetKeyState, joyx, %JoystickNumber%JoyX
}

```



# Using a Joystick as a Mouse

This script converts a joystick into a three-button mouse. It allows each button to drag just like a mouse button and it uses virtually no CPU time. Also, it will move the cursor faster depending on how far you push the joystick from center. You can personalize various settings at the top of the script.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; Increase the following value to make the mouse
cursor move faster:
```

```
JoyMultiplier := 0.30
```

```
; Decrease the following value to require less
joystick displacement-from-center
; to start moving the mouse. However, you may
need to calibrate your joystick
; -- ensuring it's properly centered -- to avoid
cursor drift. A perfectly tight
; and centered joystick could use a value of 1:
```

```
JoyThreshold := 3
```

```
; Change the following to true to invert the Y-
axis, which causes the mouse to
; move vertically in the direction opposite the
stick:
```

```
InvertYAxis := false
```

```
; Change these values to use joystick button
numbers other than 1, 2, and 3 for
; the left, right, and middle mouse buttons.
Available numbers are 1 through 32.
; Use the Joystick Test Script to find out your
```

joystick's numbers more easily.

```
ButtonLeft := 1
ButtonRight := 2
ButtonMiddle := 3
```

; If your joystick has a POV control, you can use it as a mouse wheel. The

; following value is the number of milliseconds between turns of the wheel.

; Decrease it to have the wheel turn faster:

```
WheelDelay := 250
```

; If your system has more than one joystick, increase this value to use a joystick

; other than the first:

```
JoystickNumber := 1
```

; END OF CONFIG SECTION -- Don't change anything below this point unless you want

; to alter the basic nature of the script.

```
#SingleInstance
```

```
JoystickPrefix := "%JoystickNumber%Joy"
```

```
Hotkey, %JoystickPrefix%ButtonLeft%, ButtonLeft
```

```
Hotkey, %JoystickPrefix%ButtonRight%, ButtonRight
```

```
Hotkey, %JoystickPrefix%ButtonMiddle%,
```

```
ButtonMiddle
```

; Calculate the axis displacements that are needed to start moving the cursor:

```
JoyThresholdUpper := 50 + JoyThreshold
```

```
JoyThresholdLower := 50 - JoyThreshold
```

```
if InvertYAxis
```

```
 YAxisMultiplier := -1
```

```
else
```

```
 YAxisMultiplier := 1
```

```
SetTimer, WatchJoystick, 10 ; Monitor the
movement of the joystick.
```

```
GetKeyState, JoyInfo, %JoystickNumber%JoyInfo
if Instr(JoyInfo, "P") ; Joystick has POV
control, so use it as a mouse wheel.
```

```
SetTimer, Mousewheel, %WheelDelay%
```

```
return ; End of auto-execute section.
```

```
; The subroutines below do not use KeyWait because
that would sometimes trap the
; WatchJoystick quasi-thread beneath the wait-for-
button-up thread, which would
; effectively prevent mouse-dragging with the
joystick.
```

```
ButtonLeft:
```

```
SetMouseDelay, -1 ; Makes movement smoother.
```

```
MouseClick, left,,, 1, 0, D ; Hold down the left
mouse button.
```

```
SetTimer, WaitForLeftButtonUp, 10
```

```
return
```

```
ButtonRight:
```

```
SetMouseDelay, -1 ; Makes movement smoother.
```

```
MouseClick, right,,, 1, 0, D ; Hold down the
right mouse button.
```

```
SetTimer, WaitForRightButtonUp, 10
```

```
return
```

```
ButtonMiddle:
```

```
SetMouseDelay, -1 ; Makes movement smoother.
```

```
MouseClick, middle,,, 1, 0, D ; Hold down the
right mouse button.
```

```

SetTimer, WaitForMiddleButtonUp, 10
return

WaitForLeftButtonUp:
if GetKeyState(JoystickPrefix . ButtonLeft)
 return ; The button is still, down, so keep
waiting.
; Otherwise, the button has been released.
SetTimer,, off
SetMouseDelay, -1 ; Makes movement smoother.
MouseClick, left,,, 1, 0, U ; Release the mouse
button.
return

WaitForRightButtonUp:
if GetKeyState(JoystickPrefix . ButtonRight)
 return ; The button is still, down, so keep
waiting.
; Otherwise, the button has been released.
SetTimer,, off
MouseClick, right,,, 1, 0, U ; Release the mouse
button.
return

WaitForMiddleButtonUp:
if GetKeyState(JoystickPrefix . ButtonMiddle)
 return ; The button is still, down, so keep
waiting.
; Otherwise, the button has been released.
SetTimer,, off
MouseClick, middle,,, 1, 0, U ; Release the mouse
button.
return

WatchJoystick:
MouseNeedsToBeMoved := false ; Set default.
GetKeyState, joyx, %JoystickNumber%JoyX

```

```

GetKeyState, joyx, %JoystickNumber%JoyX
if joyx > JoyThresholdUpper
{
 MouseNeedsToBeMoved := true
 DeltaX := Round(joyx - JoyThresholdUpper)
}
else if joyx < JoyThresholdLower
{
 MouseNeedsToBeMoved := true
 DeltaX := Round(joyx - JoyThresholdLower)
}
else
 DeltaX := 0
if joyy > JoyThresholdUpper
{
 MouseNeedsToBeMoved := true
 DeltaY := Round(joyy - JoyThresholdUpper)
}
else if joyy < JoyThresholdLower
{
 MouseNeedsToBeMoved := true
 DeltaY := Round(joyy - JoyThresholdLower)
}
else
 DeltaY := 0
if MouseNeedsToBeMoved
{
 SetMouseDelay, -1 ; Makes movement smoother.
 MouseMove, % DeltaX * JoyMultiplier, % DeltaY
* JoyMultiplier * YAxisMultiplier, 0, R
}
return

MouseWheel:
GetKeyState, JoyPOV, %JoystickNumber%JoyPOV
if JoyPOV = -1 ; No angle.
 return

```

```
if (JoyPOV > 31500 or JoyPOV < 4500) ; Forward
 Send {WheelUp}
else if JoyPOV >= 13500 and JoyPOV <= 22500 ;
Back
 Send {WheelDown}
return
```

# WinLIRC Client

This script receives notifications from [WinLIRC](#) whenever you press a button on your remote control. It can be used to automate Winamp, Windows Media Player, etc. It's easy to configure. For example, if WinLIRC recognizes a button named "VolUp" on your remote control, create a label named VolUp and beneath it use the command `SoundSet +5` to increase the soundcard's volume by 5%.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; Here are the steps to use this script:
; 1) Configure WinLIRC to recognize your remote
control and its buttons.
; WinLIRC is at http://winlirc.sourceforge.net
; 2) Edit the WinLIRC path, address, and port in
the CONFIG section below.
; 3) Launch this script. It will start the WinLIRC
server if needed.
; 4) Press some buttons on your remote control. A
small window will
; appear showing the name of each button as you
press it.
; 5) Configure your buttons to send keystrokes and
mouse clicks to
; windows such as Winamp, Media Player, etc.
See the examples below.
```

```
; HISTORY OF CHANGES
; July 19, 2016:
; - Revised code for AHK v2 compatibility
; March 2, 2007:
; - Improved reliability via "Critical" in
ReceiveData().
```

```
; October 5, 2005:
; - Eliminated Winsock warning dialog "10054" upon
system shutdown/logoff.
; - Added option "DelayBetweenButtonRepeats" to
throttle the repeat speed.
```

```
; -----
-
; CONFIGURATION SECTION: Set your preferences
here.
```

```
; -----
-
; Some remote controls repeat the signal rapidly
while you're holding down
; a button. This makes it difficult to get the
remote to send only a single
; signal. The following setting solves this by
ignoring repeated signals
; until the specified time has passed. 200 is
often a good setting. Set it
; to 0 to disable this feature.
```

```
DelayBetweenButtonRepeats := 200
```

```
; Specify the path to WinLIRC, such as
C:\WinLIRC\winlirc.exe
```

```
WinLIRC_Path :=
"%A_ProgramFiles%\WinLIRC\winlirc.exe"
```

```
; Specify WinLIRC's address and port. The most
common are 127.0.0.1 (localhost) and 8765.
```

```
WinLIRC_Address := "127.0.0.1"
WinLIRC_Port := "8765"
```

```
; Do not change the following two lines. Skip them
and continue below.
```

```
Gosub WinLIRC_Init
return
```

```
; -----
; ASSIGN ACTIONS TO THE BUTTONS ON YOUR REMOTE
; -----
; Configure your remote control's buttons below.
Use WinLIRC's names
; for the buttons, which can be seen in your
WinLIRC config file
; (.cf file) -- or you can press any button on
your remote and the
; script will briefly display the button's name in
a small window.
;
; Below are some examples. Feel free to revise or
delete them to suit
; your preferences.
```

```
VolUp:
SoundSet +5 ; Increase master volume by 5%. On
Vista, replace this line with: Send {Volume_Up}
return
```

```
VolDown:
SoundSet -5 ; Reduce master volume by 5%. On
Vista, replace this line with: Send {Volume_Down}
return
```

```
ChUp:
WinGetClass, ActiveClass, A
if ActiveClass =~ "(Winamp v1\\.x|Winamp PE)$" ;
Winamp is active.
 Send {right} ; Send a right-arrow keystroke.
else ; Some other type of window is active.
 Send {WheelUp} ; Rotate the mouse wheel up by
one notch.
return
```

```
ChDown:
WinGetClass, ActiveClass, A
if ActiveClass ~= "(Winamp v1\\.x|Winamp PE)$" ;
Winamp is active.
 Send {left} ; Send a left-arrow keystroke.
else ; Some other type of window is active.
 Send {WheelDown} ; Rotate the mouse wheel
down by one notch.
return
```

```
Menu:
if WinExist("Untitled - Notepad")
{
 WinActivate
}
else
{
 Run, Notepad
 WinWait, Untitled - Notepad
 WinActivate
}
Send Here are some keystrokes sent to Notepad.
{Enter}
return
```

```
; The examples above give a feel for how to
accomplish common tasks.
; To learn the basics of AutoHotkey, check out the
Quick-start Tutorial
; at http://www.autohotkey.com/docs/Tutorial.htm
```

```
; -----
; END OF CONFIGURATION SECTION
; -----
; Do not make changes below this point unless you
want to change the core
; functionality of the script.
```

```

WinLIRC_Init:
OnExit("ExitSub") ; For connection cleanup
purposes.

; Launch WinLIRC if it isn't already running:
ProcessExist, ErrorLevel, winlirc.exe
if not ErrorLevel ; No PID for WinLIRC was found.
{
 if !FileExist(WinLIRC_Path)
 {
 MsgBox The file "%WinLIRC_Path%" does not
exist. Please edit this script to specify its
location.
 ExitApp
 }
 Run %WinLIRC_Path%
 Sleep 200 ; Give WinLIRC a little time to
initialize (probably never needed, just for peace
of mind).
}

; Connect to WinLIRC (or any type of server for
that matter):
socket := ConnectToAddress(WinLIRC_Address,
WinLIRC_Port)
if socket = -1 ; Connection failed (it already
displayed the reason).
 ExitApp

; Find this script's main window:
ProcessExist, ErrorLevel ; This sets ErrorLevel
to this script's PID (it's done this way to
support compiled scripts).
DetectHiddenWindows On
ScriptMainWindowId := WinExist("ahk_class
AutoHotkey ahk_pid " . ErrorLevel)

```

DetectHiddenWindows Off

```
; When the OS notifies the script that there is
incoming data waiting to be received,
; the following causes a function to be launched
to read the data:
```

```
NotificationMsg := 0x5555 ; An arbitrary message
number, but should be greater than 0x1000.
```

```
OnMessage(NotificationMsg, "ReceiveData")
```

```
; Set up the connection to notify this script via
message whenever new data has arrived.
```

```
; This avoids the need to poll the connection and
thus cuts down on resource usage.
```

```
FD_READ := 1 ; Received when data is available
to be read.
```

```
FD_CLOSE := 32 ; Received when connection has
been closed.
```

```
if DllCall("Ws2_32\WSAAsyncSelect", "UInt",
socket, "UInt", ScriptMainWindowId, "UInt",
NotificationMsg, "Int", FD_READ|FD_CLOSE)
```

```
{
 MsgBox % "WSAAsyncSelect() indicated Winsock
error " . DllCall("Ws2_32\WSAGetLastError")
 ExitApp
}
```

```
return
```

```
ConnectToAddress(IPAddress, Port)
```

```
; This can connect to most types of TCP servers,
not just WinLIRC.
```

```
; Returns -1 (INVALID_SOCKET) upon failure or the
socket ID upon success.
```

```
{
 VarSetCapacity(wsaData, 400)
```

```

 result := DllCall("Ws2_32\WSAStartup",
"UShort", 0x0002, "UInt", &wsaData;) ; Request
Winsock 2.0 (0x0002)
; Since WSAStartup() will likely be the first
Winsock function called by this script,
; check ErrorLevel to see if the OS has
Winsock 2.0 available:
 if ErrorLevel
 {
 MsgBox WSAStartup() could not be called
due to error %ErrorLevel%. Winsock 2.0 or higher
is required.
 return -1
 }
 if result ; Non-zero, which means it failed
(most Winsock functions return 0 upon success).
 {
 MsgBox % "WSAStartup() indicated Winsock
error " . DllCall("Ws2_32\WSAGetLastError")
 return -1
 }

 AF_INET := 2
 SOCK_STREAM := 1
 IPPROTO_TCP := 6
 socket := DllCall("Ws2_32\socket", "Int",
AF_INET, "Int", SOCK_STREAM, "Int", IPPROTO_TCP)
 if socket = -1
 {
 MsgBox % "socket() indicated Winsock error
" . DllCall("Ws2_32\WSAGetLastError")
 return -1
 }

; Prepare for connection:
 SizeOfSocketAddress := 16
 VarSetCapacity(SocketAddress,

```

```

SizeOfSocketAddress)
 InsertInteger(2, SocketAddress, 0, AF_INET)
; sin_family
 InsertInteger(DllCall("ws2_32\htons",
"UShort", Port), SocketAddress, 2, 2) ; sin_port
 InsertInteger(DllCall("ws2_32\inet_addr",
"AStr", IPAddress), SocketAddress, 4, 4) ;
sin_addr.s_addr

; Attempt connection:
 if DllCall("ws2_32\connect", "UInt", socket,
"UInt", &SocketAddress;, "Int",
SizeOfSocketAddress)
 {
 MsgBox % "connect() indicated Winsock
error " . DllCall("ws2_32\WSAGetLastError") . ".
Is WinLIRC running?"
 return -1
 }
 return socket ; Indicate success by returning
a valid socket ID rather than -1.
}

```

```

ReceiveData(wParam, lParam)
; By means of OnMessage(), this function has been
set up to be called automatically whenever new
data
; arrives on the connection. It reads the data
from WinLIRC and takes appropriate action
depending
; on the contents.
{
 Critical ; Prevents another of the same
message from being discarded due to thread-
already-running.

```

```

socket := wParam
ReceivedDataSize := 4096 ; Large in case a
lot of data gets buffered due to delay in
processing previous data.

VarSetCapacity(ReceivedData, ReceivedDataSize,
0) ; 0 for last param terminates string for use
with recv().
ReceivedDataLength := DllCall("Ws2_32\recv",
"UInt", socket, "Str", ReceivedData, "Int",
ReceivedDataSize, "Int", 0)
if ReceivedDataLength = 0 ; The connection
was gracefully closed, probably due to exiting
WinLIRC.
ExitApp ; The OnExit routine will call
WSACleanup() for us.
if ReceivedDataLength = -1
{
WinsockError :=
DllCall("Ws2_32\WSAGetLastError")
if WinsockError = 10035 ; WSAEWOULDBLOCK,
which means "no more data to be read".
return 1
if WinsockError <> 10054 ; WSAECONNRESET,
which happens when WinLIRC closes via system
shutdown/logoff.
; Since it's an unexpected error,
report it. Also exit to avoid infinite loop.
MsgBox % "recv() indicated Winsock
error " . WinsockError
ExitApp ; The OnExit routine will call
WSACleanup() for us.
}
; Otherwise, process the data received.
Testing shows that it's possible to get more than
one line
; at a time (even for explicitly-sent IR

```

```

signals), which the following method handles
properly.
 ; Data received from WinLIRC looks like the
following example (see the WinLIRC docs for
details):
 ; 0000000000eab154 00 NameOfButton
NameOfRemote
 Loop, parse, %ReceivedData%, `n, `r
 {
 if A_LoopField ~= "^(|BEGIN|SIGHUP|END)$"
; Ignore blank lines and WinLIRC's start-up
messages.
 continue
 ButtonName := "" ; Init to blank in case
there are less than 3 fields found below.
 Loop, parse, %A_LoopField%, %A_Space% ;
Extract the button name, which is the third field.
 if A_Index = 3
 ButtonName := A_LoopField
 global DelayBetweenButtonRepeats ;
Declare globals to make them available to this
function.
 static PrevButtonName, PrevButtonTime,
RepeatCount ; These variables remember their
values between calls.
 if (ButtonName != PrevButtonName ||
A_TickCount - PrevButtonTime >
DelayBetweenButtonRepeats)
 {
 if IsLabel(ButtonName) ; There is a
subroutine associated with this button.
 Gosub %ButtonName% ; Launch the
subroutine.
 else ; Since there is no associated
subroutine, briefly display which button was
pressed.
 }

```

```

 if (ButtonName == PrevButtonName)
 RepeatCount += 1
 else
 RepeatCount := 1
 ToolTip, Button from WinLIRC,
%ButtonName% (%RepeatCount%)
 SetTimer, SplashOff, -3000 ; This
allows more signals to be processed while
displaying the window.
 }
 PrevButtonName := ButtonName
 PrevButtonTime := A_TickCount
 }
}
return 1 ; Tell the program that no further
processing of this message is needed.
}

```

```

SplashOff:
ToolTip
return

```

```

InsertInteger(pInteger, ByRef pDest, pOffset := 0,
pSize := 4)
; The caller must ensure that pDest has sufficient
capacity. To preserve any existing contents in
pDest,
; only pSize number of bytes starting at pOffset
are altered in it.
{
 Loop pSize ; Copy each byte in the integer
into the structure as raw binary data.
 DllCall("RtlFillMemory", "UInt", &pDest; +

```

```
pOffset + A_Index-1, "UInt", 1, "UChar", pInteger
>> 8*(A_Index-1) & 0xFF)
}
```

```
ExitSub() ; This function is called automatically
when the script exits for any reason.
```

```
{
 ; MSDN: "Any sockets open when WSACleanup is
called are reset and automatically
 ; deallocated as if closesocket was called."
 DllCall("Ws2_32\WSACleanup")
 ExitApp
}
```

# Script Performance

The following commands may affect performance depending on the nature of the script: `SendMode`, `SetKeyDelay`, `SetMouseDelay`, `SetWinDelay`, `SetControlDelay`, and `SetDefaultMouseSpeed`.

## Built-in Performance Features

Each script is semi-compiled while it is being loaded and syntax-checked. In addition to reducing the memory consumed by the script, this also greatly improves runtime performance.

Here are the technical details of the optimization process (semi-compiling):

- Input and output variables (when their names don't contain references to other variables) and `group` names are resolved to memory addresses.
- `Loops`, `blocks`, `IFs`, and `ELSEs` are given the memory addresses of their related jump-points in the script.
- The destination of each `Hotkey`, `Gosub`, and `Goto` is resolved to a memory address unless it is a variable.
- Each command name is replaced by an address in a jump table.
- Each line is pre-parsed into a list of parameters, and each parameter is pre-parsed into a list of `variables` (if any).
- Each `expression` is tokenized and converted from infix to postfix.
- Each reference to a `variable` or `function` is resolved to a memory address.
- Literal integers in expressions are replaced with binary integers.

In addition, during script execution, binary numbers are cached in variables to avoid conversions to/from strings.

# OnEvent

Registers a function or method to be called when the given event is raised by a GUI window or control.

```
Object.OnEvent(EventName, Callback [, AddRemove := 1])
```

## Parameters

### Object

A [Gui](#) or [GuiControl](#) object.

### EventName

The name of the event.

### Callback

The function, method or object to call when the event is raised.

If this parameter is a string, its meaning depends on whether the GUI has an [event sink](#) (that is, whether [GuiCreate](#)'s *EventObj* parameter was specified). If the GUI has an event sink, the string must be the name of a method belonging to the event sink; otherwise, it must be the name of a function.

To register a function regardless of whether the GUI has an event sink, pass a [function reference](#).

## AddRemove

One of the following values:

**1** (the default): Call the callback after any previously registered callbacks.

**-1**: Call the callback before any previously registered callbacks.

**0**: Do not call the callback.

## Callback Parameters

If the callback is a method registered by name, its hidden *this* parameter seamlessly receives the event sink object (that is, the object to which the method belongs). This parameter is not shown in the parameter lists in this documentation.

Since *Callback* can be an object, it can be a [BoundFunc object](#) which inserts additional parameters at the beginning of the parameter list and then calls another function. This is a general technique not specific to OnEvent, so is generally ignored by the rest of this documentation.

The callback's first explicit parameter is always *Object*; i.e. the [Gui](#) or [GuiControl](#) object which raised the event.

Many events pass additional parameters about the event, as described for each event.

As with all methods or functions called dynamically, the callback is not required to declare parameters which the callback itself does not need. If an event has more parameters than are declared by the callback, they will simply be ignored (unless the callback is [variadic](#)).

The callback can declare more parameters than the event provides if (and only if) the additional parameters are declared optional. However, the use of optional parameters is not recommended as future versions of the program may extend an event with additional parameters, in which case the optional parameters would

stop receiving their default values.

## Callback Return Value

If multiple callbacks have been registered for an event, a callback may return a non-empty value to prevent any remaining callbacks from being called.

The return value may have additional meaning for specific events. For example, a `Close` callback may return a non-zero number (such as `true`) to prevent the GUI window from closing.

## Callback Name

By convention, the syntax of each event below is shown with a function name of the form `|ObjectType■EventName|`, for clarity. Scripts are not required to follow this convention, and can use any valid function name.

## Threads

Each event callback is called in a new [thread](#), and therefore starts off fresh with the default values for settings such as [SendMode](#). These defaults can be changed in the [auto-execute section](#).

Whenever a GUI [thread](#) is launched, that thread's [last found window](#) starts off as the GUI window itself. This allows functions for windows and controls -- such as [WinGetStyle](#), [WinSetTransparent](#), and [ControlGetFocus](#) -- to omit [WinTitle](#) and [WinText](#) when operating upon the GUI window itself (even if it is hidden).

Except where noted, each event is limited to one thread at a time, per object. If an event is raised before a previous thread started by that event finishes, it is usually discarded. To prevent this, use [Critical](#) as the callback's first line (however, this will also buffer/defer other [threads](#) such as the press of a hotkey).

## Destroying the GUI

When a GUI is destroyed, all event callbacks are released. Therefore, if the GUI is destroyed while an event is being dispatched, subsequent event callbacks are not called. For clarity, callbacks should [return a non-empty value](#) after destroying the GUI.

## Events

The following events are supported by `Gui` objects:

| Event                    | Raised when...                                                                |
|--------------------------|-------------------------------------------------------------------------------|
| <code>Close</code>       | The window is closed.                                                         |
| <code>ContextMenu</code> | The user right-clicks within the window or presses the Apps key or Shift-F10. |
| <code>DropFiles</code>   | Files/folders are dragged and dropped onto the window.                        |
| <code>Escape</code>      | The user presses the Escape key while the GUI window is active.               |
| <code>Size</code>        | The window is resized, minimized, maximized or restored.                      |

The following events are supported by `GuiControl` objects, depending on the control type:

| Event                    | Raised when...                                                                                                   |
|--------------------------|------------------------------------------------------------------------------------------------------------------|
| <code>Change</code>      | The control's value changes.                                                                                     |
| <code>Click</code>       | The control is clicked.                                                                                          |
| <code>DoubleClick</code> | The control is double-clicked.                                                                                   |
| <code>ColClick</code>    | One of the <code>ListView</code> 's column headers is clicked.                                                   |
| <code>ContextMenu</code> | The user right-clicks the control or presses the Apps key or Shift-F10 while the control has the keyboard focus. |
| <code>Focus</code>       | The control gains the keyboard focus.                                                                            |
| <code>LoseFocus</code>   | The control loses the keyboard focus.                                                                            |
| <code>ItemCheck</code>   | A <code>ListView</code> or <code>TreeView</code> item is checked or unchecked.                                   |
| <code>ItemEdit</code>    | A <code>ListView</code> or <code>TreeView</code> item's label is edited by the user.                             |

|            |                                                                            |
|------------|----------------------------------------------------------------------------|
| ItemExpand | A TreeView item is expanded or collapsed.                                  |
| ItemFocus  | The focused item changes in a ListView.                                    |
| ItemSelect | A ListView or TreeView item is selected, or a ListView item is deselected. |

## Window Events

### Close

Launched when the user or another program attempts to close the window, such as by pressing the X button in its title bar, selecting "Close" from its system menu, or calling `WinClose`.

```
Gui_Close(GuiObj)
```

By default, the window is automatically hidden after the callback returns, or if no callbacks were registered. A callback can prevent this by returning 1 (or `true`), which will also prevent any remaining callbacks from being called. The callback can hide the window immediately by calling `Gui.Hide()`, or destroy the window by calling `Gui.Destroy()`.

For example, this GUI shows a confirmation prompt before closing:

```
myGui := GuiCreate()
myGui.AddText("", "Press Alt+F4 or the X button
in the title bar.")
myGui.OnEvent("close", "myGui_Close")
myGui_Close(thisGui) { ; Declaring this
parameter is optional.
 if MsgBox("Are you sure you want to close
the GUI?",, "y/n") = "No"
 return true ; true = 1
}
myGui.Show()
```

## ContextMenu

Launched whenever the user right-clicks anywhere in the window except the title bar and menu bar. It is also launched in response to pressing the Apps key or Shift-F10.

```
Gui_ContextMenu(GuiObj, GuiCtrlObj, Item,
IsRightClick, X, Y)
```

### GuiCtrlObj

The [GuiControl object](#) of the control that received the event (blank if none).

### Item

When a [ListBox](#), [ListView](#), or [TreeView](#) is the target of the context menu (as determined by *GuiCtrlObj*), *Item* specifies which of the control's items is the target.

**ListBox:** The number of the currently focused row. Note that a standard [ListBox](#) does not focus an item when it is right-clicked, so this might not be the clicked item.

**ListView** and **TreeView:** For right-clicks, *Item* contains the clicked item's ID or row number (or 0 if the user clicked somewhere other than an item). For the [AppsKey](#) and [Shift-F10](#), *Item* contains the selected item's ID or row number.

## IsRightClick

`True` if the user clicked the right mouse button.

`False` if the user pressed the Apps key or Shift-F10.

## X, Y

The X and Y coordinates of where the script should display the menu (e.g. `Menu, MyContext, Show, %X%, %Y%`). Coordinates are relative to the upper-left corner of the window's client area.

Unlike most other GUI events, the `ContextMenu` event can have more than one concurrent [thread](#).

Each control can have its own `ContextMenu` event callback which is called before any callback registered for the Gui object. Control-specific callbacks omit the `GuiObj` parameter, but all other parameters are the same.

Note: Since `Edit` and `MonthCal` controls have their own context menu, a right-click in one of them will not launch the `ContextMenu` event.

## DropFiles

Launched whenever files/folders are dropped onto the window as part of a drag-and-drop operation (but if this callback is already running, drop events are ignored).

```
Gui_DropFiles(GuiObj, GuiCtrlObj, FileArray, X, Y)
```

## GuiCtrlObj

The `GuiControl` object of the control upon which the files were dropped (blank if none).

## FileArray

An array (object) of filenames, where `FileArray[1]` is the first file and `FileArray.Length()` returns the number of files. A `for-loop` can be used to iterate through the files:

```
Gui_DropFiles(GuiObj, GuiCtrlObj,
FileArray, X, Y) {
 for i, file in FileArray
 MsgBox File %i% is:`n%file%
}
```

## X, Y

The X and Y coordinates of where the files were dropped, relative to the upper-left corner of the window's client area.

## Escape

Launched when the user presses the Escape key while the GUI window is active.

```
Gui_Escape(GuiObj)
```

By default, pressing the Escape key has no effect. Known limitation: If the first control in the window is disabled (possibly depending on control type), the

Escape event will not be launched. There may be other circumstances that produce this effect.

## Size

Launched when the window is resized, minimized, maximized, or restored.

```
Gui_Size(GuiObj, MinMax, Width, Height)
```

### MinMax

One of the following values:

0: The window is neither minimized nor maximized.

1: The window is maximized.

-1: The window is minimized.

Note that a maximized window can be resized without restoring/un-maximizing it, so a value of 1 does not necessarily mean that this event was raised in response to the user maximizing the window.

### Width, Height

The new width and height of the window's client area, which is the area excluding title bar, menu bar, and borders.

A script may use the Size event to reposition and resize controls in response to the user's resizing of the window.

When the window is resized (even by the script), the Size event might not be

raised immediately. As with other window events, if the current thread is `uninterruptible`, the Size event won't be raised until the thread becomes interruptible. If the script has just resized the window, follow this example to ensure the Size event is raised immediately:

```
Critical Off ; Even if Critical On was never
used.
Sleep -1
```

`Gui.Show()` automatically does a `Sleep -1`, so it is generally not necessary to call `Sleep` in that case.

## Control Events

### Change

Raised when the control's value changes.

```
Ctrl_Change(GuiCtrlObj, Info)
```

#### Info

For [Slider](#) controls, *Info* is a numeric value indicating how the slider moved. For details, see [Detecting Changes \(Slider\)](#).

For all other controls, *Info* currently has no meaning.

To retrieve the control's new value, use [GuiCtrlObj.Value](#).

Applies to: [DDL](#), [ComboBox](#), [ListBox](#), [Edit](#), [DateTime](#), [MonthCal](#), [Hotkey](#), [UpDown](#), [Slider](#), [Tab](#).

### Click

Raised when the control is clicked.

```
Ctrl_Click(GuiCtrlObj, Info)
Link_Click(GuiCtrlObj, Info, Href)
```

#### Info

ListView: The row number of the clicked item, or 0 if the mouse was not over an item.

TreeView: The ID of the clicked item, or 0 if the mouse was not over an item.

Link: The link's ID attribute (a string) if it has one, otherwise the link's index (an integer).

StatusBar: The part number of the clicked section (however, the part number might be a very large integer if the user clicks near the sizing grip at the right side of the bar).

For all other controls, *Info* currently has no meaning.

## Href

Link: The link's HREF attribute. Note that if a Click event callback is registered, the HREF attribute is not automatically executed.

Applies to: [Text](#), [Pic](#), [Button](#), [CheckBox](#), [Radio](#), [ListView](#), [TreeView](#), [Link](#), [StatusBar](#).

## DoubleClick

Raised when the control is double-clicked.

```
Ctrl_DoubleClick(GuiCtrlObj, Info)
```

## Info

ListView, TreeView and StatusBar: Same as for the [Click](#) event.

ListBox: The position of the focused item. Double-clicking empty space below the last item usually focuses the last item and leaves the selection as it was.

Applies to: [Text](#), [Pic](#), [Button](#), [CheckBox](#), [Radio](#), [ComboBox](#), [ListBox](#), [ListView](#), [TreeView](#), [StatusBar](#).

## ColClick

Raised when one of the ListView's column headers is clicked.

```
Ctrl_ ColClick(GuiCtrlObj, Info)
```

## Info

The one-based column number that was clicked. This is the original number assigned when the column was created; that is, it does not reflect any dragging and dropping of columns done by the user.

Applies to: [ListView](#).

## ContextMenu

Raised when the user right-clicks the control or presses the Apps key or Shift-F10 while the control has the keyboard focus.

```
Ctrl_ContextMenu(GuiCtrlObj, Item, IsRightClick, X,
Y)
```

For details, see [ContextMenu](#).

Applies to: All controls except Edit and MonthCal (and the Edit control within a ComboBox), which have their own standard context menu.

## Focus / LoseFocus

Raised when the control gains or loses the keyboard focus.

```
Ctrl_Focus(GuiCtrlObj, Info)
Ctrl_LoseFocus(GuiCtrlObj, Info)
```

### Info

Reserved.

Applies to: [Button](#), [CheckBox](#), [Radio](#), [DDL](#), [ComboBox](#), [ListBox](#), [ListView](#), [TreeView](#), [Edit](#), [DateTime](#).

Not supported: [Hotkey](#), [Slider](#), [Tab](#) and [Link](#). Note that [Text](#), [Pic](#), [MonthCal](#), [UpDown](#) and [StatusBar](#) controls do not accept the keyboard focus.

## ItemCheck

Raised when a ListView or TreeView item is checked or unchecked.

```
Ctrl_ ItemCheck(GuiCtrlObj, Item, Checked)
```

Applies to: [ListView](#), [TreeView](#).

## ItemEdit

Raised when a [ListView](#) or [TreeView](#) item's label is edited by the user.

```
Ctrl_ ItemEdit(GuiCtrlObj, Item)
```

An item's label can only be edited if `-ReadOnly` has been used in the control's options.

Applies to: [ListView](#), [TreeView](#).

## ItemExpand

Raised when a [TreeView](#) item is expanded or collapsed.

```
Ctrl_ ItemExpand(GuiCtrlObj, Item, Expanded)
```

Applies to: [TreeView](#).

## ItemFocus

Raised when the focused item changes in a [ListView](#).

```
Ctrl_ ItemFocus(GuiCtrlObj, Item)
```

---

Applies to: [ListView](#).

## ItemSelect

Raised when a [ListView](#) or [TreeView](#) item is selected, or a [ListView](#) item is deselected.

```
ListView_ItemSelect(GuiCtrlObj, Item, Selected)
TreeView_ItemSelect(GuiCtrlObj, Item)
```

Applies to: [ListView](#), [TreeView](#).

[ListView](#): This event is raised once for each item being deselected or selected, so can be raised multiple times in response to a single action by the user.

## Other Events

Other types of GUI events can be detected and acted upon via [OnNotify](#), [OnCommand](#) or [OnMessage](#). For example, a script can display context-sensitive help via ToolTip whenever the user moves the mouse over particular controls in the window. This is demonstrated in the [GUI ToolTip example](#).

# Var := expression

Evaluates an expression and stores the result in a [variable](#).

```
Var := expression
```

## Parameters

### Var

The name of the [variable](#) in which to store the result of *expression*.

### Expression

See [expressions](#) and the examples below for details.

Note that this parameter is **not optional**. To empty the variable, specify an empty string. For example: `x := ""`

## Remarks

The := operator is optimized so that it performs just as quickly as the = operator for simple cases such as the following:

```
x := y ; Same performance as x = %y%
x := 5 ; Same performance as x = 5.
x := "literal string" ; Same performance as x
= literal string.
```

The words `true` and `false` are built-in constants containing 1 and 0. They

can be used to make a script more readable as in these examples:

```
CaseSensitive := false
ContinueSearch := true
```

It is possible to create a [pseudo-array](#) with this command and any others that accept an *OutputVar*. This is done by making *OutputVar* contain a reference to another variable, e.g. `Array%i% := Var/100 + 5`. See [Arrays](#) for more information.

## Related

[Expressions](#), [IF \(expression\)](#), [Functions](#), [EnvSet](#), [Arrays](#)

## Examples

```
Var := 3
Var := "literal string"
Var := Price * (1 - Discount/100)

Finished := not Done or A_Index > 100
if not Finished
{
 FileAppend, %NewText%\n, %TargetFile%
 return
}
else
 ExitApp
```

# Arrays

## Associative Arrays

Self-contained associative arrays can be created by calling `Object`. For example:

```
; Create the array, initially empty:
Array := Object()

; Write to the array:
Loop, Read, C:\Guest List.txt ; This loop
retrieves each line from the file, one at a
time.
{
 Array.Push(A_LoopReadLine) ; Append this
line to the array.
}

; Read from the array:
; Loop, Array.Length() ; More traditional
approach.
for index, element in Array ; Recommended
approach in most cases.
{
 ; Using "Loop", indices must be consecutive
numbers from 1 to the number
 ; of elements in the array (or they must be
calculated within the loop).
 ; MsgBox % "Element number " . A_Index . "
is " . Array[A_Index]
 ; Using "for", both the index (or "key")
and its associated value
 ; are provided, and the index can be *any*
```

value of your choosing.

```
 MsgBox % "Element number " . index . " is "
 . element
}
```

This shows only a small subset of the [functionality](#) provided by [objects](#). Items can be set, retrieved, inserted, removed and enumerated. Strings and objects can be used as keys in addition to numbers. Objects can be stored as values in other objects and passed as function parameters or return values. Objects can also be [extended](#) with new functionality.

Though Push() and for-loops have their uses, some users might find it easier to use the more traditional approach:

**; Each array must be initialized before use:**

```
Array := Object()
```

**; Array%j% := A\_LoopField**

```
Array[j] := A_LoopField
```

**; Array%j%\_%k% := A\_LoopReadLine**

```
Array[j, k] := A_LoopReadLine
```

```
ArrayCount := 0
```

```
Loop, Read, C:\Guest List.txt
```

```
{
```

```
 ArrayCount += 1
```

**; Array%ArrayCount% := A\_LoopReadLine**

```
 Array[ArrayCount] := A_LoopReadLine
```

```
}
```

```
Loop ArrayCount
```

```
{
```

```
; element := Array%A_Index%
 element := Array[A_Index]
; MsgBox % "Element number " . A_Index . "
is " . Array%A_Index%
 MsgBox % "Element number " . A_Index . "
is " . Array[A_Index]
}
```

*ArrayCount* is left as a variable for convenience, but can be stored in the array itself with `Array.Count := n` or it can be removed and `Array.Length` used in its place.

## Pseudo-Arrays

Pseudo-arrays are mostly conceptual: Each array is really just a collection of sequentially numbered [variables](#) or [functions](#), each one being perceived as an *element* of the array. AutoHotkey does not link these elements together in any way.

In addition to array-creating commands like [StrSplit](#) and "[WinGet List](#)", any command that accepts an `OutputVar` or that assigns a value to a variable can be used to create an array. The simplest example is the [assignment operator](#) (`:=`), as shown below:

```
Array%j% := A_LoopField
```

Multidimensional arrays are possible by using a separator character of your choice between the indices. For example:

```
Array%j%_%k% := A_LoopReadLine
```

The following example demonstrates how to create and access an array, in this case a series of names retrieved from a text file:

```
; Write to the array:
ArrayCount := 0
Loop, Read, C:\Guest List.txt ; This loop
retrieves each line from the file, one at a
time.
{
```

```

 ArrayCount += 1 ; Keep track of how many
items are in the array.
 Array%ArrayCount% := A_LoopReadLine ;
Store this line in the next array element.
}

; Read from the array:
Loop ArrayCount
{
 ; The following line uses the := operator
to retrieve an array element:
 element := Array%A_Index% ; A_Index is a
built-in variable.
 ; Alternatively, you could use the "% "
prefix to make MsgBox or some other command
expression-capable:
 MsgBox % "Element number " . A_Index . " is
" . Array%A_Index%
}

```

A concept related to arrays is the use of [NumPut](#) and [NumGet](#) to store/retrieve a collection of numbers in binary format. This might be helpful in cases where performance and/or memory conservation are important.

# Value is Type

Within an expression, checks whether a value is of a given type or is numeric, uppercase, etc.

```
if Value is Type
if !(Value is Type)
```

## Parameters

### Value

An [expression](#) which produces the value to check.

### Type

An [expression](#) which produces a type string or object, as described below.

## Type Strings

Type strings are case-insensitive.

|         |                                                                                                                                                                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| integer | True if <i>Value</i> is an integer or a purely numeric string (decimal or hexadecimal) without a decimal point. Leading and trailing spaces and tabs are allowed. The string may start with a plus or minus sign and must not be empty.             |
| float   | True if <i>Value</i> is a floating point number or a purely numeric string containing a decimal point. Leading and trailing spaces and tabs are allowed. The string may start with a plus sign, minus sign, or decimal point and must not be empty. |

|         |                                                                                                                                                                                                                                                                                                                                             |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| number  | True if <code>Value</code> is "integer" or <code>Value</code> is "float" is true.                                                                                                                                                                                                                                                           |
| object  | True if <code>Value</code> is an object.                                                                                                                                                                                                                                                                                                    |
| byref   | True if <code>Value</code> is a reference to a <code>ByRef</code> parameter and the function's caller passed a variable reference.                                                                                                                                                                                                          |
| Strings |                                                                                                                                                                                                                                                                                                                                             |
| digit   | True if <code>Value</code> is an integer, an empty string, or a string which contains only the characters 0 through 9. Other characters such as the following are not allowed: spaces, tabs, plus signs, minus signs, decimal points, hexadecimal digits, and the 0x prefix.                                                                |
| xdigit  | Hexadecimal digit: Same as <code>digit</code> except the characters A through F (uppercase or lowercase) are also allowed. A prefix of 0x is tolerated if present.                                                                                                                                                                          |
| alpha   | True if <code>Value</code> is a string and is empty or contains only alphabetic characters. False if there are any digits, spaces, tabs, punctuation, or other non-alphabetic characters anywhere in the string. For example, if <code>Value</code> contains a space followed by a letter, it is <i>not</i> considered to be <i>alpha</i> . |
| upper   | True if <code>Value</code> is a string and is empty or contains only uppercase characters. False if there are any digits, spaces, tabs, punctuation, or other non-uppercase characters anywhere in the string.                                                                                                                              |
| lower   | True if <code>Value</code> is a string and is empty or contains only lowercase characters. False if there are any digits, spaces, tabs, punctuation, or other non-lowercase characters anywhere in the string.                                                                                                                              |
| alnum   | Same as <i>alpha</i> except that integers and characters 0 through 9 are also allowed.                                                                                                                                                                                                                                                      |
| space   | True if <code>Value</code> is a string and is empty or contains only whitespace consisting of the following characters: space ( <code>%A_Space%</code> ), tab ( <code>%A_Tab%</code> or <code>`t</code> ), linefeed ( <code>`n</code> ), return ( <code>`r</code> ), vertical tab ( <code>`v</code> ), and formfeed ( <code>`f</code> ).    |
|         | True if <code>Value</code> is a valid date-time stamp, which can be all or just                                                                                                                                                                                                                                                             |

time

the leading part of the [YYYYMMDDHH24MISS](#) format. For example, a 4-digit string such as 2004 is considered valid. Use [StrLen](#) to determine whether additional time components are present.

Years less than 1601 are not considered valid because the operating system generally does not support them. The maximum year considered valid is 9999.

The word DATE may be used as a substitute for the word TIME, with the same result.

## Objects

If *Type* is an [object](#), the result is true if *Value* is an object derived from *Type*, directly or indirectly. For example:

```
x := {}
y := new x ; Equivalent to y := {base: x}
z := new y
MsgBox(x is x) ; False
MsgBox(y is x) ; True
MsgBox(z is x) ; True
```

## Remarks

[is](#) can be used anywhere an expression is expected. [Precedence rules](#) apply, so for instance, [x or y is z](#) is the same as [x or \(y is z\)](#).

To check for multiple types, use [is](#) multiple times. For example, [x is "integer" or x is "space"](#).

Since literal numbers such as `128`, `0x7F` and `1.0` are converted to pure numbers before the script begins executing, the format of the literal number is lost. Consequently, type checks act on the decimal form of the number. For example, `0x7F is "digit"` is equivalent to `"128" is "digit"`.

The system locale is ignored unless [StringCaseSense Locale](#) has been used.

## Related

[A\\_YYYY](#), [FileGetTime](#), [if \(expression\)](#), [StrLen](#), [InStr](#), [StrUpper](#), [DateAdd](#)

## Example

```
if var is "float"
 MsgBox, %var% is a floating point number.
else if var is 'integer'
 MsgBox, %var% is an integer.
if var is "time"
 MsgBox, %var% is also a valid date-time.
```

# Clipboard and ClipboardAll

*Clipboard* is a built-in variable that reflects the current contents of the Windows clipboard if those contents can be expressed as text. By contrast, *ClipboardAll()* returns an object containing everything on the clipboard, such as pictures and formatting.

Each line of text on *Clipboard* typically ends with carriage return and linefeed (CR+LF), which can be expressed in the script as `r`n`. Files (such as those copied from an open Explorer window via Control-C) are considered to be text: They are automatically converted to their filenames (with full path) whenever *Clipboard* is referenced in the script. To extract the files one by one, follow this example:

```
Loop, parse, %clipboard%, `n, `r
{
 Result := MsgBox("File number %A_Index% is
%A_LoopField%.`n`nContinue?", , 4)
 if Result = "No", break
}
```

To arrange the filenames in alphabetical order, use the `Sort` command. To write the filenames on the clipboard to a file, use `FileAppend, %clipboard%`r`n, C:\My File.txt`. To change how long the script will keep trying to open the clipboard -- such as when it is in use by another application -- use `#ClipboardTimeOut`.

## Basic examples:

```
clipboard := "my text" ; Give the clipboard
entirely new contents.
clipboard := "" ; Empty the clipboard.
clipboard := clipboard ; Convert any copied
files, HTML, or other formatted text to plain
text.
clipboard := "%clipboard% Text to append." ;
Append some text to the clipboard.
StrReplace, clipboard, %clipboard%, ABC, DEF
; Replace all occurrences of ABC with DEF (also
converts the clipboard to plain text).
```

### Using ClipWait to improve script reliability:

```
clipboard := "" ; Start off empty to allow
ClipWait to detect when the text has arrived.
Send ^c
ClipWait ; Wait for the clipboard to contain
text.
MsgBox Control-C copied the following contents
to the clipboard:`n`n%clipboard%
```

## ClipboardAll (saving and restoring everything on the clipboard)

Creates an object containing everything on the clipboard (such as pictures and formatting).

```
ClipSaved := ClipboardAll([Data, Size])
```

### Parameters

Omit both parameters to retrieve the current contents of the clipboard. Otherwise, specify one or both parameters to create an object containing the given binary clipboard data.

#### Data

A string containing binary data, or a pure integer which is the address of the binary data. The data must be in a specific format, so typically originates from a previous call to ClipboardAll(). See FileAppend below.

#### Size

The number of bytes of data to use. This is optional when *Data* is a string.

### ClipboardAll Object

The return value is a ClipboardAll object, which has three properties:

#### Data

A string containing raw binary data which represents the clipboard

contents. This is typically passed to `FileAppend` or `File.RawWrite` to write it to file.

### Ptr

The address of the data contained by the object. This address is valid until the object is freed.

### Size

The size, in bytes, of the raw binary data.

## Saving and Restoring the Clipboard

`ClipboardAll` contains everything on the clipboard (such as pictures and formatting). It is most commonly used to save the clipboard's contents so that the script can temporarily use the clipboard for an operation. When the operation is completed, the script restores the original clipboard contents as shown below:

```
ClipSaved := ClipboardAll() ; Save the entire
clipboard to a variable of your choice.
; ... here make temporary use of the clipboard,
such as for quickly pasting large amounts of
text ...
Clipboard := ClipSaved ; Restore the original
clipboard. Note the use of Clipboard (not
ClipboardAll).
ClipSaved := "" ; Free the memory in case the
clipboard was very large.
```

`ClipboardAll` may also be saved to a file:

```
; Option 1: Delete any existing file and then
use FileAppend in "RAW" mode.
```

```
FileDelete "C:\Company Logo.clip"
FileAppend ClipboardAll().Data, "C:\Company
Logo.clip", "RAW" ; The file extension does not
matter.
```

```
; Option 2: Use FileOpen in overwrite mode and
File.RawWrite.
```

```
FileOpen("C:\Company Logo.clip",
"w").RawWrite(ClipboardAll().Data)
```

To later load the file back onto the clipboard (or into a variable), follow this example:

```
Clipboard := ClipboardAll(FileRead("C:\Company
Logo.clip", "RAW"))
```

## Notes

If *ClipboardAll* cannot retrieve one or more of the data objects (formats) on the clipboard, they will be omitted but all the remaining objects will be stored.

A variable containing clipboard data can be copied to another variable as in this example: `ClipSaved2 := ClipSaved`.

`ClipWait` may be used to detect when the clipboard contains data (optionally including non-text data).

Binary data returned by the `Data` property internally consists of a four-byte format type, followed by a four-byte data-block size, followed by the data-block

for that format. If the clipboard contained more than one format (which is almost always the case), these three items are repeated until all the formats are included. The data ends with a four-byte format type of 0.

Known limitation: Retrieving *ClipboardAll* while cells from Microsoft Excel are on the clipboard may cause Excel to display a "no printers" dialog.

Clipboard utilities written in AutoHotkey v1:

- Deluxe Clipboard: Provides unlimited number of private, named clipboards to Copy, Cut, Paste, Append or CutAppend of selected text.  
[www.autohotkey.com/forum/topic2665.html](http://www.autohotkey.com/forum/topic2665.html)
- ClipStep: Control multiple clipboards using only the keyboard's Ctrl-X-C-V. [www.autohotkey.com/forum/topic4836.html](http://www.autohotkey.com/forum/topic4836.html)

## OnClipboardChange

Scripts can detect changes to the content of the Clipboard by using [OnClipboardChange](#).

# ErrorLevel

This is a built-in variable that is set to indicate the success or failure of some of the commands (not all commands change the value of ErrorLevel). A value of 0 usually indicates success, and any other value usually indicates failure. You can also set the value of ErrorLevel yourself.

Of special interest is that `RunWait` sets ErrorLevel to be the exit code of the program it ran. Most programs yield an exit code of zero if they completed successfully.

Each `thread` starts with an ErrorLevel of 0 and retains its own value of ErrorLevel, meaning that if the `current thread` is interrupted by another, when the original thread is resumed it will still have its original value of ErrorLevel, not the ErrorLevel that may have been set by the interrupting thread.

Although ErrorLevel typically contains a number (most often 0 or 1), the script can assign other types of values to it.

Note: Since some commands set ErrorLevel to values higher than 1, it is best not check whether ErrorLevel is 1, but instead whether ErrorLevel is not zero.

## Example

```
WinWait, MyWindow, , 1
if ErrorLevel > 0 ; i.e. it's not blank or zero.
 MsgBox, The window does not exist.
else
```



# Labels

A label identifies a line of code, and can be used as a [Goto](#) target or to form a [subroutine](#).

```
LabelName:
```

To create a label, write the label name followed by a colon as shown above. Aside from whitespace and comments, no other code can be written on the same line.

**Names:** Label names are not case sensitive, and may consist of any characters other than space, tab, comma and the [escape character](#) (```). However, due to style conventions, it is generally better to use only letters, numbers, and the underscore character (for example: *MyListView*, *Menu\_File\_Open*, and *outer\_loop*).

**Scope:** Each function has its own list of local labels. If a local label has the same name as a global label, the local label takes precedence (and the global label is inaccessible). Some limitations apply when [using subroutines within a function](#).

**Target:** The target of a label is the next line of executable code. Executable code includes commands, assignments, [expressions](#) and [blocks](#), but not directives, labels, hotkeys or hotstrings. In the following example, `run_notepad` and `#n` both point at the `Run` line:

```
run_notepad:
```

```
#n::
 Run Notepad
 return
```

**Execution:** Like directives, labels have no effect when reached during normal execution. In the following example, a message box is shown twice - once during execution of the subroutine by `Gosub`, and again after the subroutine returns:

```
gosub Label1

Label1:
MsgBox %A_ThisLabel%
return
```

## Subroutines

A subroutine is a portion of code which can be *called* to perform a specific task. Execution of a subroutine begins at the target of a label and continues until a **Return** or **Exit** is encountered. Since the end of a subroutine depends on flow of control, any label can act as both a Goto target and the beginning of a subroutine.

## Dynamic Labels

Many commands which accept a label name also accept a [variable](#) reference such as %MyLabel%, in which case the name stored in the variable is used as the target. However, performance is slightly reduced because the target label must be "looked up" each time rather than only once when the script is first loaded.

## Hotkeys and Hotstrings

Hotkey and hotstring labels are also valid targets for Goto, Gosub and other commands. However, if a hotkey or hotstring has multiple variants, the variant closest to the top of the script is used. All of the hotkey's modifiers or hotstring's options are also part of its label name, but the final double-colon (::) is omitted.

## Named Loops

A label can also be used to identify a loop for the [Continue](#) and [Break](#) commands. This allows the script to easily continue or break out of any number of nested loops.

## Functions

Functions can be used in place of labels in a number of cases, including:

- Hotkey
- Menu
- SetTimer

The benefits of functions are that they can use local variables, and in some cases (such as Gui control events) they also accept parameters containing useful information.

## Related

[IsLabel\(\)](#), [A\\_ThisLabel](#), [Gosub](#), [Goto](#), [SetTimer](#), [Hotkey](#)

# Language Codes

The following list contains the language name that corresponds to each language code that can be contained in the `A_Language` variable. The language code itself is the last four digits on the left side of the equal sign below. For example, if `A_Language` contains 0436, the system's default language is Afrikaans. Note: Codes that contain letters might use either uppercase or lowercase.

You can compare `A_Language` directly to one or more of the 4-digit codes below; for example: `if (A_Language = "0436")`. Alternatively, you can paste the entire list into a script and then access the name of the current language as demonstrated at the bottom of the list.

```
languageCode_0436 := "Afrikaans"
languageCode_041c := "Albanian"
languageCode_0401 := "Arabic_Saudi_Arabia"
languageCode_0801 := "Arabic_Iraq"
languageCode_0c01 := "Arabic_Egypt"
languageCode_0401 := "Arabic_Saudi_Arabia"
languageCode_0801 := "Arabic_Iraq"
languageCode_0c01 := "Arabic_Egypt"
languageCode_1001 := "Arabic_Libya"
languageCode_1401 := "Arabic_Algeria"
languageCode_1801 := "Arabic_Morocco"
languageCode_1c01 := "Arabic_Tunisia"
languageCode_2001 := "Arabic_Oman"
languageCode_2401 := "Arabic_Yemen"
languageCode_2801 := "Arabic_Syria"
languageCode_2c01 := "Arabic_Jordan"
languageCode_3001 := "Arabic_Lebanon"
languageCode_3401 := "Arabic_Kuwait"
```







```
the_language := languageCode_%A_Language% ; Get
the name of the system's default language.
MsgBox %the_language% ; Display the language
name.
```

# Regular Expression Callouts

Callouts provide a means of temporarily passing control to the script in the middle of regular expression pattern matching. For detailed information about the PCRE-standard callout feature, see [pcre.txt](#).

Callouts are currently supported only by [RegExMatch](#) and [RegExReplace](#).

## Syntax

The syntax for a callout in AutoHotkey is `(?CNumber:Function)`, where both *Number* and *Function* are optional. Colon ':' is allowed only if *Function* is specified, and is optional if *Number* is omitted. If *Function* is specified but is not the name of a user-defined function, a compile error occurs and pattern-matching does not begin.

If *Function* is omitted, the function name must be specified in a variable named `pcre_callout`. If both a global variable and local variable exist with this name, the local variable takes precedence. If `pcre_callout` does not contain the name of a user-defined function, callouts which omit *Function* are ignored.

## Callout Functions

```
MyFunction(Match, CalloutNumber, FoundPos, Haystack,
NeedleRegEx)
{
 ...
}
```

```
}
```

Callout functions may define up to 5 parameters:

- **Match:** Equivalent to the *UnquotedOutputVar* of `RegexMatch`, including the creation of array variables if appropriate.
- **CalloutNumber:** Receives the *Number* of the callout.
- **FoundPos:** Receives the position of the current potential match.
- **Haystack:** Receives the *Haystack* passed to `RegexMatch` or `RegexReplace`.
- **NeedleRegex:** Receives the *NeedleRegex* passed to `RegexMatch` or `RegexReplace`.

These names are suggestive only. Actual names may vary.

Pattern-matching may proceed or fail depending on the return value of the callout function:

- If the function returns **0** or does not return a numeric value, matching proceeds as normal.
- If the function returns **1** or greater, matching fails at the current point, but the testing of other matching possibilities goes ahead.
- If the function returns **-1**, matching is abandoned.
- If the function returns a value less than **-1**, it is treated as a PCRE error code and matching is abandoned. `RegexMatch` returns a blank string, while `RegexReplace` returns the original *Haystack*. In either case, `ErrorLevel` contains the error code.

For example:

```
Haystack := "The quick brown fox jumps over the
lazy dog."
RegexMatch(Haystack, "i)(The) (\w+)\b(?
CCallout)")
Callout(m) {
 MsgBox
m[0]=%m[0]%\`nm[1]=%m[1]%\`nm[2]=%m[2]%\`
 return 1
}
```

In the above example, *Func* is called once for each substring which matches the part of the pattern preceding the callout. **\b** is used to exclude incomplete words in matches such as *The quic*, *The qui*, *The qu*, etc.

## EventInfo

Additional information is available by accessing the `pcre_callout_block` structure via **A\_EventInfo**.

```
version := NumGet(A_EventInfo, 0,
"Int")
callout_number := NumGet(A_EventInfo, 4,
"Int")
offset_vector := NumGet(A_EventInfo, 8)
subject := NumGet(A_EventInfo, 8 +
A_PtrSize)
subject_length := NumGet(A_EventInfo, 8 +
A_PtrSize*2, "Int")
start_match := NumGet(A_EventInfo, 12 +
A_PtrSize*2, "Int")
```

```

current_position := NumGet(A_EventInfo, 16 +
A_PtrSize*2, "Int")
capture_top := NumGet(A_EventInfo, 20 +
A_PtrSize*2, "Int")
capture_last := NumGet(A_EventInfo, 24 +
A_PtrSize*2, "Int")
pad := A_PtrSize=8 ? 4 : 0 ; Compensate for
64-bit data alignment.
callout_data := NumGet(A_EventInfo, 28 +
pad + A_PtrSize*2)
pattern_position := NumGet(A_EventInfo, 28 +
pad + A_PtrSize*3, "Int")
next_item_length := NumGet(A_EventInfo, 32 +
pad + A_PtrSize*3, "Int")
if version >= 2
 mark := StrGet(NumGet(A_EventInfo, 36 +
pad + A_PtrSize*3, "Int"), "UTF-8")

```

For more information, see [pcre.txt](#), [NumGet](#) and [A\\_PtrSize](#).

## Auto-Callout

Including **C** in the options of the pattern enables the auto-callout mode. In this mode, callouts equivalent to **(?C255)** are inserted before each item in the pattern. For example, the following template may be used to debug regular expressions:

```

; Set the default callout function.
pcre_callout := "DebugRegEx"

; Call RegExMatch with auto-callout option C.
RegExMatch("xxxabc123xyz", "C)abc.*xyz")

```

```

DebugRegEx(Match, CalloutNumber, FoundPos,
Haystack, NeedleRegEx)
{
 ; See pcre.txt for descriptions of these
 fields.
 start_match := NumGet(A_EventInfo, 12
+ A_PtrSize*2, "Int")
 current_position := NumGet(A_EventInfo, 16
+ A_PtrSize*2, "Int")
 pad := A_PtrSize=8 ? 4 : 0
 pattern_position := NumGet(A_EventInfo, 28
+ pad + A_PtrSize*3, "Int")
 next_item_length := NumGet(A_EventInfo, 32
+ pad + A_PtrSize*3, "Int")

 ; Point out >>current match<<.
 _HAYSTACK:=SubStr(Haystack, 1, start_match)
 . ">>" SubStr(Haystack, start_match +
1, current_position - start_match)
 . "<<" SubStr(Haystack,
current_position + 1)

 ; Point out >>next item to be evaluated<<.
 _NEEDLE:= SubStr(NeedleRegEx, 1,
pattern_position)
 . ">>" SubStr(NeedleRegEx,
pattern_position + 1, next_item_length)
 . "<<" SubStr(NeedleRegEx,
pattern_position + 1 + next_item_length)

 ListVars
 ; Press Pause to continue.
 Pause
}

```

## Remarks

Callouts are executed on the current quasi-thread, but the previous value of `A_EventInfo` will be restored after the callout function returns. `ErrorLevel` is not set until immediately before `RegexMatch` or `RegexReplace` returns.

PCRE is optimized to abort early in some cases if it can determine that a match is not possible. For all callouts to be called in such cases, it may be necessary to disable these optimizations by specifying `( *NO_START_OPT )` at the start of the pattern.

# FileExist() / DirExist()

Checks for the existence of a file or folder and returns its attributes.

```
AttributeString := FileExist(FilePattern)
AttributeString := DirExist(FilePattern)
```

## Parameters

### FilePattern

The path, filename, or file pattern to check. *FilePattern* is assumed to be in %A\_WorkingDir% if an absolute path isn't specified.

## Return Value

Both functions return an attribute string (a subset of "RASHNDOCT"):

R = READONLY

A = ARCHIVE

S = SYSTEM

H = HIDDEN

N = NORMAL

D = DIRECTORY

O = OFFLINE

C = COMPRESSED

T = TEMPORARY

FileExist returns the attribute string of the first matching file or folder, "X" if the file has no attributes (rare), or an empty string if no file or folder is found.

DirExist returns the attribute string of the first matching folder, or an empty string if no folder is found.

## Remarks

FileExist searches for both files and folders, whereas DirExist searches only for folders. `InStr(FileExist(P), "D")` returns zero (false) if the first matching file is not a folder, even if a matching folder exists.

Unlike [FileGetAttrib](#), FileExist supports wildcard patterns and always returns a non-empty value if a matching file exists.

Since an empty string is seen as "false", the function's return value can always be used as a quasi-boolean value. For example, the statement `If FileExist("C:\My File.txt")` would be true if the file exists and false otherwise.

## Related

[FileGetAttrib](#), [File-loops](#)

## Examples

```
if FileExist("D:\")
 MsgBox, The drive exists.
if FileExist("D:\Docs*.txt")
```

```
 MsgBox, At least one .txt file exists.
if !FileExist("C:\Temp\FlagFile.txt")
 MsgBox, The target file does not exist.
```

**; The following example shows how to check the file for a particular attribute:**

```
if InStr(FileExist("C:\My File.txt"), "H")
 MsgBox The file is hidden.
```

# Func Object

Represents a user-defined or built-in function which can be called by the script.

For information about other objects which can be called like functions, see [Function Objects](#).

A reference to a Func object is also known as a *function reference*. To retrieve a function reference, use the Func function as in the following example:

```
; Retrieve a reference to the function named
"StrLen".
fn := Func("StrLen")

; Display information about the function.
MsgBox % fn.Name "()" is " (fn.IsBuiltIn ?
"built-in." : "user-defined.")
```

If *fn* is a function reference, `Func(fn)` returns it. Thus, `fn := Func(fn)` can be used to ensure *fn* is a function reference. If *fn* is neither a valid function name nor a function reference, *Func* returns a blank value.

# Call

Calls the function.

```
Func.call(Parameters)
%Func%(Parameters)
```

The second form (a [dynamic function call](#)) also works with function names (strings) and other kinds of [function objects](#).

Parameters and return value are defined by the function.

# Bind

Binds parameters to the function and returns a [BoundFunc object](#).

```
BoundFunc := Func.Bind(Parameters)
```

*Parameters* can be any number of parameters.

For details and examples, see [BoundFunc object](#).

## **Name**

Returns the function's name.

```
Func . Name
```

## IsBuiltIn

Returns *true* if the function is built-in and *false* otherwise.

```
Func.IsBuiltIn
```

## IsVariadic

Returns *true* if the function is *variadic* and *false* otherwise.

```
Func.IsVariadic
```

## MinParams

Returns the number of required parameters.

```
Func.MinParams
```

## MaxParams

Returns the number of formally-declared parameters for a user-defined function or maximum parameters for a built-in function.

```
Func.MaxParams
```

If the function is [variadic](#), the return value indicates the maximum number of parameters which can be accepted by the function without overflowing into the "variadic\*" parameter.

## IsByRef()

Determines whether a parameter is ByRef.

```
Func.IsByRef(ParamIndex)
```

**ParamIndex** Optional: the one-based index of a parameter. If omitted, the return value indicates whether the function has any ByRef parameters.

**Returns** An empty string if the function is built-in or *ParamIndex* is invalid; otherwise, a boolean value indicating whether the parameter is ByRef.

## IsOptional()

Determines whether a parameter is optional.

```
Func.IsOptional(ParamIndex)
```

**ParamIndex** Optional: the one-based index of a parameter. If omitted, the return value indicates whether the function has any optional parameters.

**Returns** An empty string if *ParamIndex* is invalid; otherwise, a boolean value indicating whether the parameter is optional.

Parameters do not need to be formally declared if the function is variadic.

Built-in functions are supported.

# ObjBindMethod()

Creates a [BoundFunc](#) object which calls a method of a given object.

```
BoundFunc := ObjBindMethod(Obj, Method, Params)
```

## Parameters

### Obj

Any object.

### Method

A method name.

### *Params*

Any number of parameters.

## Remarks

For details and examples, see [BoundFunc](#) object.

# Context Sensitive Help in Any Editor -- by Rajat

This script makes Ctrl+2 (or another hotkey of your choice) show the help file page for the selected AutoHotkey command or keyword. If nothing is selected, the command name will be extracted from the beginning of the current line.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; The hotkey below uses the clipboard to provide
compatibility with the maximum
; number of editors (since ControlGet doesn't work
with most advanced editors).
; It restores the original clipboard contents
afterward, but as plain text,
; which seems better than nothing.
```

```
$^2::
```

```
; The following values are in effect only for the
duration of this hotkey thread.
; Therefore, there is no need to change them back
to their original values
; because that is done automatically when the
thread ends:
```

```
SetWinDelay 10
SetKeyDelay 0
```

```
C_ClipboardPrev := clipboard
clipboard := ""
```

```
; Use the highlighted word if there is one (since
sometimes the user might
; intentionally highlight something that isn't a
command):
```

```

Send, ^c
ClipWait, 0.1
if ErrorLevel <> 0
{
 ; Get the entire line because editors treat
 ; cursor navigation keys differently:
 Send, {home}+{end}^c
 ClipWait, 0.2
 if ErrorLevel <> 0 ; Rare, so no error is
 reported.
 {
 clipboard := C_ClipboardPrev
 return
 }
}
C_Cmd := Trim(clipboard) ; This will trim leading
and trailing tabs & spaces.
clipboard := C_ClipboardPrev ; Restore the
original clipboard for the user.
Loop, parse, %C_Cmd%, %A_Space%, ; The first
space or comma is the end of the command.
{
 C_Cmd := A_LoopField
 break ; i.e. we only need one iteration.
}
if !WinExist("AutoHotkey Help")
{
 ; Determine AutoHotkey's location:
 RegRead, ahk_dir,
 HKEY_LOCAL_MACHINE\SOFTWARE\AutoHotkey, InstallDir
 if ErrorLevel ; Not found, so look for it in
 some other common locations.
 {
 if A_AhkPath
 SplitPath, %A_AhkPath%,, ahk_dir
 else if FileExist("../..\AutoHotkey.chm")
 ahk_dir := "../.."
 }
}

```

```
 else if
FileExist("%A_ProgramFiles%\AutoHotkey\AutoHotkey.
chm")
 ahk_dir :=
"%A_ProgramFiles%\AutoHotkey"
 else
 {
 MsgBox Could not find the AutoHotkey
folder.
 return
 }
 }
 Run %ahk_dir%\AutoHotkey.chm
 WinWait AutoHotkey Help
}
```

**; The above has set the "last found" window which we use below:**

```
WinActivate
WinWaitActive
StrReplace, C_Cmd, %C_Cmd%, #, {#}
send, !n{home}+{end}%C_Cmd%{enter}
return
```

# Automating Winamp

This section demonstrates how to control Winamp via hotkey even when it is minimized or inactive. This information has been tested with Winamp 2.78c but should work for other major releases as well. Please post changes and improvements in the forum or contact the author.

This example makes the Ctrl+Alt+P hotkey equivalent to pressing Winamp's pause/unpause button:

```
^!p::
if !WinExist("ahk_class Winamp v1.x")
 return
; Otherwise, the above has set the "last found"
window for use below.
ControlSend, ahk_parent, c ; Pause/Unpause
return
```

Here are some of the keyboard shortcuts available in Winamp 2.x (may work in other versions too). The above example can be revised to use any of these keys:

| Key to send | Effect                       |
|-------------|------------------------------|
| c           | Pause/UnPause                |
| x           | Play/Restart/UnPause         |
| v           | Stop                         |
| +v          | Stop with Fadeout            |
| ^v          | Stop after the current track |

|         |                        |
|---------|------------------------|
| b       | Next Track             |
| z       | Previous Track         |
| {left}  | Rewind 5 seconds       |
| {right} | Fast-forward 5 seconds |
| {up}    | Turn Volume Up         |
| {down}  | Turn Volume Down       |

```
; This next example asks Winamp which track
number is currently active:
SendMessage, 1024, 0, 120, ahk_class Winamp
V1.x
if ErrorLevel <> "ERROR"
{
 ErrorLevel += 1 ; Winamp's count starts at
0, so adjust by 1.
 MsgBox, Track #%ErrorLevel% is active or
playing.
}
```

# Changing MsgBox's Button Names

This is a working example script that uses a timer to change the names of the buttons in a MsgBox dialog. Although the button names are changed, the MsgBox's return value still requires that the buttons be referred to by their original names.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
#SingleInstance
SetTimer, ChangeButtonNames, 50
Result := MsgBox("Choose a button:", "Add or
Delete", 4)
if Result = "Yes"
 MsgBox, You chose Add.
else
 MsgBox, You chose Delete.
return

ChangeButtonNames:
if !WinExist("Add or Delete")
 return ; Keep waiting.
SetTimer,, off
WinActivate
ControlSetText, Button1, &Add;
ControlSetText, Button2, &Delete;
return
```

# Numpad 000 Key

This example script makes the special 000 key that appears on certain keypads into an equals key. You can change the action by replacing the `Send, =` line with line(s) of your choice.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
#MaxThreadsPerHotkey 5 ; Allow multiple threads
for this hotkey.
$Numpad0::
#MaxThreadsPerHotkey 1
; Above: Use the $ to force the hook to be used,
which prevents an
; infinite loop since this subroutine itself sends
Numpad0, which
; would otherwise result in a recursive call to
itself.
DelayBetweenKeys := 30 ; Adjust this value if it
doesn't work.
if A_PriorHotkey = A_ThisHotkey
{
 if A_TimeSincePriorHotkey < DelayBetweenKeys
 {
 if Numpad0Count = ""
 Numpad0Count := 2 ; i.e. This one plus
the prior one.
 else if Numpad0Count = 0
 Numpad0Count := 2
 else
 {
 ; Since we're here, Numpad0Count must
be 2 as set by
; prior calls, which means this is the
```

```

third time the
; the key has been pressed. Thus, the
hotkey sequence
; should fire:
Numpad0Count := 0
Send, = ; ***** This is the action
for the 000 key
}
; In all the above cases, we return
without further action:
CalledReentrantly := true
return
}
}
; Otherwise, this Numpad0 event is either the
first in the series
; or it happened too long after the first one
(e.g. perhaps the
; user is holding down the Numpad0 key to auto-
repeat it, which
; we want to allow). Therefore, after a short
delay -- during
; which another Numpad0 hotkey event may re-
entrantly call this
; subroutine -- we'll send the key on through if
no reentrant
; calls occurred:
Numpad0Count := 0
CalledReentrantly := false
; During this sleep, this subroutine may be
reentrantly called
; (i.e. a simultaneous "thread" which runs in
parallel to the
; call we're in now):
Sleep, %DelayBetweenKeys%
if CalledReentrantly = true ; Another "thread"
changed the value.

```

```
{
 ; Since it was called reentrantly, this key
 event was the first in
 ; the sequence so should be suppressed (hidden
 from the system):
 CalledReentrantly := false
 return
}
; Otherwise it's not part of the sequence so we
send it through normally.
; In other words, the *real* Numpad0 key has been
pressed, so we want it
; to have its normal effect:
Send, {Numpad0}
return
```

# GetEnv

Add environment variables to script.

## GetEnv

|          |          |          |
|----------|----------|----------|
| Command  | Example: | GetEnv   |
| Function | Example: | GetEnv() |

## Examples

```
GetEnv
ListVars
Pause
```

# ExeThread

Create a real additional AutoHotkey thread in current process without using AutoHotkey.dll (based on [NewThread](#)).

```
OutputVar := ExeThread(Script [, Parameters, Title, Wait])
```

**Command Example:** ExeThread "MsgBox Message from thread."

**Function Example:** **Thread** := ExeThread("MsgBox Message from thread.")

## Parameters

### OutputVar

The name of the variable in which to store the Thread object for newly created thread.

### Script

The AutoHotkey script to execute as string or variable containing a string. If you want to start a script from file simply use [#Include](#) directive. If you want to start empty script use [#Persistent](#).

### Parameters (optional)

Command line parameters that will be available in built-in variable [A\\_Args](#) object.

## Title (optional)

Title to use for new script.

## Wait (optional)

Milliseconds to wait for new script to start.

## General Remarks

### Methods

|                                 |                                                                                |
|---------------------------------|--------------------------------------------------------------------------------|
| <a href="#">ahkFunction</a>     | Call a function via SendMessage method.                                        |
| <a href="#">ahkPostFunction</a> | Call a function via PostMessage method (does not wait until function returns). |
| <a href="#">ahkExecuteLine</a>  | Executes script from given line pointer.                                       |
| <a href="#">ahkLabel</a>        | Goto (PostMessage) or Gosub (SendMessage) a Label.                             |
| <a href="#">ahkFindFunction</a> | Find a function and return its pointer.                                        |
| <a href="#">ahkFindLabel</a>    | Find a label and return its pointer.                                           |
| <a href="#">addFile</a>         | Add and optionally execute additional script/code from file.                   |
| <a href="#">addScript</a>       | Add and optionally execute additional script/code from text/memory/variable.   |
| <a href="#">ahkExec</a>         | Execute some script/code from text/memory/variable temporarily.                |
| <a href="#">ahkassign</a>       | Assign a value to variable or pointer of variable.                             |
| <a href="#">ahkgetvar</a>       | Retrieve a value from a variable.                                              |
| <a href="#">ahkPause</a>        | Pause Script.                                                                  |

## Related

[AhkThread](#), [NewThread](#), [ThreadObj](#), [Objects](#), [DllCall](#)

## Examples



# ThreadObj

Create a real additional AutoHotkey thread in current process without using AutoHotkey.dll (based on [NewThread](#)).

```
OutputVar := ThreadObj(Script [, Parameters, Title])
```

```
Function Example: Thread := ThreadObj("MsgBox
Message from thread.")
```

## Parameters

### OutputVar

The name of the variable in which to store the Thread object for newly created thread.

### Script

The AutoHotkey script to execute as string or variable containing a string. If you want to start a script from file simply use [#Include](#) directive.

### Parameters (optional)

Command line parameters that will be available in built-in variable [A\\_Args](#) object.

### Title (optional)

Title to use for new script.

## Methods

```
MsgBox % thread.variable ; get a variable value
thread.variable:=Value ; set a variable value
ReturnValue := thread.Call("FunctionName",
parameters...)
hread.PostCall("FunctionName",
parameters...)
thread.FuncPtr("FunctionName")
thread.LabelPtr("LabelName")
thread.Exec("MsgBox Script")
thread.AddFile("FilePath", Execute)
thread.AddScript("MsgBox Script", Execute)
thread.GotoLabel("Label")
thread.GoSubLabel("Label")
thread.ExitApp()
```

## Related

[AhkThread](#), [NewThread](#), [ExeThread](#) , [Objects](#), [DllCall](#)

## Examples

```
Thread:=ThreadObj("Msgbox `%
variable:=`"Thread`"")
MsgBox % thread.variable
```

# NewThread

Create a real additional AutoHotkey thread in current process using [Thread Local Storage](#) (without using AutoHotkey.dll).

The main difference to [AutoHotkey.dll](#) thread is that a TLS thread shares the [hook](#) between all threads and uses one pool of [Hotkeys](#) and [Hotstrings](#), so if you have same hotkey in main thread and TLS thread you will get a duplicate Hotkey error.

```
OutputVar := NewThread(Script [, Parameters, Title])
```

```
Command Example: NewThread "MsgBox Message from thread."
```

```
Function Example: ThreadID := NewThread("MsgBox Message from thread.")
```

## Parameters

### OutputVar

The name of the variable in which to store the ThreadID for newly created thread.

### Script

The AutoHotkey script to execute as string or variable containing a string. If you want to start a script from file simply use [#include](#) directive.

### Parameters (optional)

Command line parameters that will be available in built-in variable `A_Args` object.

### Title (optional)

Title to use for new script.

### Related

[AhkThread](#), [ExeThread](#), [ThreadObj](#), [Objects](#), [DllCall](#)

### Examples

```
NewThread("Msgbox `% variable:=`"Thread`")
```

# Alias

Convert a local variable to an alias to represent a different variable for example in another thread.

**Alias** VariableOrName, VariableOrPointer

```
Command Example: Alias, "MyVar",
VariableOrPointer
Function Example: Alias("MyVar",
VariableOrPointer)
```

## Parameters

### VariableOrName

The name of Variable to convert to alias or the variable itself.

### VariableOrPointer

A variable or a pointer to the variable to refer to.

## Related

[NumGet](#), [DllCall](#), [VarSetCapacity](#)

## Example

```
var:="Hello World!"
Alias(refvar, var)
```

```
MsgBox % refvar
refvar:="AutoHotkey"
MsgBox % var
```

**; Same example using variable pointer**

```
var:="Hello World!"
Alias(refvar, getvar(var))
MsgBox % refvar
refvar:="AutoHotkey"
MsgBox % var
```

# BinToHex

Convert binary memory to hex string.

```
OutputVar := BinToHex(Address, Length)
```

```
Function Example: hex := BinToHex(&var;, 10)
```

## Parameters

### OutputVar

The name of the variable in which to store the hex string.

### Address

Address of variable or pointer to memory.

### Length

Length of memory in bytes to convert.

## Related

[DllCall](#), [#DllImport](#), [DynaCall](#), [WinApi](#)

## Examples

```
MsgBox % BinToHex("&123",6) ; unicode string to
hex
```

```
VarSetCapacity(var,11)
StrPut("ABC",&var;,"CP0")
MsgBox % BinToHex(var,4) ; ansi string to hex
Loop 10
 NumPut(A_Index,&var;,A_Index-1,"Char")
MsgBox % BinToHex(&var;,10) ; memory to hex
```

# BinRun

Run a executable file (.exe) from memory.

```
OutputVar := BinRun(AddressOrPath [,
CommandLineParams, ScriptCommandLineParams, Hide,
ExeToUse])
```

```
Command Example: BinRun &MyExe;, "`nMsgBox `%
A_Args.a", {a:"Hello World!"}
Function Example: PID := BinRun(&MyExe;,
"`nMsgBox `% A_Args.a", {a:"Hello World!"})
```

## Parameters

### OutputVar

The name of the variable in which to store the process identifier (PID) of new process.

### AddressOrPath

The address of a variable or a pointer to memory of the executable file (.exe).

This can be also a filepath or resource name, when both resource and filepath exist, resource takes precedence.

### CommandLineParams (optional)

Command line parameters for the executable file (.exe).

This parameter can contain the text which will be loaded through pipe. For example: "`nMsgBox `% A\_AhkVersion"

### ScriptCommandLineParams (optional)

Command line parameters for the script. This parameter can be an object as well. The object will be passed to new executable via [ObjDump](#) / [ObjLoad](#).

### Hide (optional)

True/1 to start process hidden otherwise 0/false/NULL.

### ExeToUse

Executable to use as base for new process.

By default the executable of current process is taken and `%A_WinDir%\Microsoft.NET\Framework\v2.0.50727\vbc.exe` to launch 32-bit exe from 64-bit process.

## Examples

```
FileRead,file,*c %A_AhkPath%
BinRun(&file;,"`nMsgBox `% A_Args.a",{a:"Hello
World!"})

; same as above
BinRun(A_AhkPath,"`nMsgBox `% A_Args.a",{a:"Hello
World!"})
```

# CreateScript

Creates a script from main script that can be passed to [AutoHotkey.dll](#), [BinRun](#) or [DynaRun](#).

```
OutputVar := CreateScript(Script [, Password])
```

```
Function Example: MyScript :=
CreateScript("a:=1`nStart:End`nfun{}")
```

## Parameters

### OutputVar

The name of the variable in which to store the created script.

### Script

Label or function in main Script to include.

Syntax for including label is **tartingLabelName:EndingLabelName**

Syntax for including function is **FunctionName{}**

### Password (optional)

Password used to compile in Ahk2Exe.

## General Remarks

You can have any code between included labels and functions. Script is

also loaded from Resource if A\_IsCompiled.

## Related

[AutoHotkey.dll](#), [BinRun](#), [DynaRun](#)

## Examples

```
DynaRun(CreateScript("
(
Gui, Add, Button, g
```

```
 func{}
)"), "Dynamic script")
 ExitApp
 :
 Return
 func(){
Return "test trough pipe"
 }
```

# CriticalSection

Included function to create and initialize a [Critical Section Object](#).

```
OutputVar := CriticalSection()
```

```
Function Example: CriSec := CriticalSection()
```

## Parameters

### OutputVar

The name of the variable in which to store the pointer to the Critical Section Object.

## General Remarks

A critical section object is useful in a multi-threaded environment ([AutoHotkey.dll](#)).

Critical Section Object provides synchronization similar to that provided by a mutex object but is used only by the threads of single process.

When Critical Section is owned by one thread and another thread tries to take ownership it will be halted and only continue once the thread released ownership.

To take ownership call [EnterCriticalSection\(CriSec\)](#), to release ownership call [LeaveCriticalSection\(CriSec\)](#). You can also call

[TryEnterCriticalSection\(CriSec\)](#) if you don't want to lock current thread

so it can still process messages, use Hotkeys, Timers...

## Related

[CriticalSection](#), [AutoHotkey.dll](#)

## Examples

```
CriSec:=CriticalSection()
ahkdll:=AhkThread("CriSec:=" CriSec "`nLoop
5`nEnterCriticalSection(CriSec),MsgBox(`"Critical
Section is now owned by
AutoHotkey.dll`"),LeaveCriticalSection(CriSec),Slee
p(10)")
Loop 5
 EnterCriticalSection(CriSec),MsgBox("Criti
cal Section is now owned by
AutoHotkey.exe"),LeaveCriticalSection(CriSec),Slee
p(10)
```

# DirGetParent

Retrieve parent directory for a file or folder.

```
OutputVar := DirGetParent(Path [, ParentCount])
```

```
Function Example: ParentDir :=
DirGetParent(A_AhkPath)
```

## Parameters

### OutputVar

The name of the variable in which to store the parent directory.

### Path

Path to a directory or File, the path must exist.

### ParentCount (optional)

By default (ParentCount = 1) parent directory is retrieved, to retrieve parent of parent use 2 and so on.

## General Remarks

If there is no parent directory, empty string will be returned.

## Examples



# ErrorMessage

Retrieve the error message from [system message table](#) resources.

```
OutputVar := ErrorMessage(MessageId)
```

```
Function Example: Error :=
ErrorMessage(A_LastError)
```

## Parameters

### OutputVar

The name of the variable in which to store the error message.

### MessageID

The message identifier (A\_LastError).

## Examples

```
MsgBox % ErrorMessage(87)
```

# ExtractIconFromExecutable

Extracts an icon from executable (dll or exe).

```
OutputVar := ExtractIconFromExecutable(FilePath,
IconNumber, Width, Height)
```

```
Function Example: hIcon :=
ExtractIconFromExecutable(A_AhkPath, 1, 16, 16)
```

## Parameters

### OutputVar

The name of the variable in which to store the icon handle.

### FilePath

Path to the executable file (dll or exe).

### IconNumber

The number of icon to extract.

### Width

Width of icon.

### Height

Height of icon.

## Related

[ResGet](#), [ResExist](#), [ResDelete](#), [ResDllCreate](#)

## Example

```
Gui,+LastFound
DetectHiddenWindows,On
hIcon := ExtractIconFromExecutable(A_AhkPath,2,16,
16)
SendMessage, 0x80, 0,% hIcon
SendMessage, 0x80, 1,% hIcon
Gui,Show, w400 h300
```

# FindFunc

Finds a function in currently executed script and returns a pointer to it.

```
OutputVar := FindFunc(FuncName)
```

```
Function Example: MyFuncPtr :=
FindFunc("MyFunc")
```

## Parameters

### OutputVar

The name of the variable to store the function pointer in or 0 if the function was not found.

### FuncName

The name of the function to find.

## Examples

```
; AHK Structures
global _AHKDerefType := "LPTSTR marker, {_AHKVar
*var, _AHKFunc *func, _AHKDerefType *next, UInt
symbol}, BYTE type, param_count, WORD length"
global _AHKExprTokenType := "{__int64
value_int64, double value_double, struct{{PTR
*object, _AHKDerefType *deref, _AHKVar *var, LPTSTR
marker}, {LPTSTR buf, size_t marker_length}}}, UINT
symbol, {_AHKExprTokenType *circuit_token, LPTSTR
```





# FindLabel

Finds a label in currently executed script and returns a pointer to it.

```
OutputVar := FindLabel(LabelName)
```

```
Function Example: MyLabelPtr :=
FindLabel("MyLabel")
```

## Parameters

### OutputVar

The name of the variable to store the label pointer in or 0 if the label was not found.

### LabelName

The name of the label to find.

## Examples

```
; AHK Structures
global _AHKDerefType := "LPTSTR marker, {_AHKVar
*var, _AHKFunc *func, _AHKDerefType *next, UInt
symbol}, BYTE type, param_count, WORD length"
global _AHKExprTokenType := "{__int64
value_int64, double value_double, struct{{PTR
*object, _AHKDerefType *deref, _AHKVar *var, LPTSTR
marker}, {LPTSTR buf, size_t marker_length}}, UINT
symbol, {_AHKExprTokenType *circuit_token, LPTSTR
```



# HexToBin

Convert hex string to binary memory.

```
OutputVar := HexToBin(Bin, Hex)
```

```
Function Example: address :=
HexToBin(bin, "101010")
```

## Parameters

### OutputVar

The name of the variable in which to store address of bin variable.

### Bin

Name of variable in which to store binary data.

### Hex

Hex string to convert.

## Related

[DllCall](#), [#DllImport](#), [DynaCall](#), [WinApi](#)

## Examples

```
VarSetCapacity(v, 8, 0)
```

```
NumPut(0x10101010,&v;,"INT64")
hex:=BinToHex(&v;,8)
MsgBox % hex
HexToBin(bin,hex)
MsgBox % format("0x{1:X}",NumGet(&b;,"INT64"))
```

# HIBYTE

Retrieves the high-order byte from the given value.

```
OutputVar := HIBYTE(Value)
```

```
Function Example: Byte := HIBYTE(5130)
```

## Parameters

### OutputVar

The name of the variable to store the high-order byte.

### Value

The value to convert to high-order byte.

## Examples

```
var:=5130
MsgBox % HIBYTE(var)
```

# HIWORD

Retrieves the high-order word from the given value.

```
OutputVar := HIWORD(Value)
```

```
Function Example: Word := HIWORD(1310730)
```

## Parameters

### OutputVar

The name of the variable to store the high-order word.

### Value

The value to convert to high-order word.

## Examples

```
var:=1310730
MsgBox % HIWORD(var)
```

# LoadPicture

Loads a picture from file and returns a bitmap or icon handle.

```
Handle := LoadPicture(Filename [, Options, ByRef ImageType])
```

## Parameters

### Filename

The filename of the picture, which is usually assumed to be in `A_WorkingDir` if an absolute path isn't specified. If the name of a DLL or EXE file is given without a path, it may be loaded from the directory of the current executable (AutoHotkey.exe or a compiled script) or a system directory.

### Options

A string of zero or more of the following options, with each separated from the last by a space or tab:

**Wn** and **Hn**: The width and height to load the image at, where *n* is an integer. If one dimension is omitted or -1, it is calculated automatically based on the other dimension, preserving aspect ratio. If both are omitted, the image's original size is used. If either dimension is 0, the original size is used for that dimension. For example: `"w80 h50"`, `"w48 h-1"` or `"w48"` (preserve aspect ratio), `"h0 w100"` (use original height but

override width).

**Iconn**: Indicates which icon to load from a file with multiple icons (generally an EXE or DLL file). If *n* is non-zero, the file must contain an icon. For example, `"Icon2"` loads the file's second icon.

**GDI+**: Use GDI+ to load the image, if available. For example, `"GDI+w100"`.

## ImageType

The unquoted name of a variable in which to store an integer representing the type of handle which is returned: 0 (IMAGE\_BITMAP), 1 (IMAGE\_ICON) or 2 (IMAGE\_CURSOR). If omitted or not a variable, the return value is always a bitmap handle (icons/cursors are converted if necessary).

## Remarks

LoadPicture also supports [the handle syntax](#), such as for creating a resized image based on an icon or bitmap which has already been loaded into memory, or converting an icon to a bitmap by omitting *ImageType*.

If the image needs to be freed from memory, call whichever function is appropriate for the type of handle.

```
if (not ImageType) ; IMAGE_BITMAP (0) or the
ImageType parameter was omitted.
 DllCall("DeleteObject", "ptr", Handle)
else if (ImageType = 1) ; IMAGE_ICON
```

```
DllCall("DestroyIcon", "ptr", Handle)
else if (ImageType = 2) ; IMAGE_CURSOR
 DllCall("DestroyCursor", "ptr", Handle)
```

## Related

[Image Handles](#)

## Example

```
; Pre-load and reuse some images.

Pics := []
; Find some pictures to display.
LoopFiles("%A_WinDir%\Web\Wallpaper*.jpg", "R")
{
 ; Load each picture and add it to the array.
 Pics.Push(LoadPicture(A_LoopFileFullPath))
}
if !Pics.Length()
{
 ; If this happens, edit the path on the Loop
line above.
 MsgBox("No pictures found! Try a different
 directory.")
 ExitApp()
}
; Add the picture control, preserving the aspect
ratio of the first picture.
Gui := GuiCreate()
Pic := Gui.Add("Pic", "w600 h-1 vPic +Border",
"HBITMAP:*" Pics.1)
Gui.OnEvent("Escape", "Gui_Escape")
Gui.Show()
```

```
Loop
{
 ; Switch pictures!
 Pic.Value := "HBITMAP:*" Pics[Mod(A_Index,
Pics.Length()+1)]
 Sleep(3000)
}

Gui_Escape() {
 ExitApp()
}
```

# LOBYTE

Retrieves the low-order byte from the given value.

```
OutputVar := LOBYTE(Value)
```

```
Function Example: Byte := LOBYTE(5130)
```

## Parameters

### OutputVar

The name of the variable to store the low-order byte.

### Value

The value to convert to low-order byte.

## Examples

```
var:=5130
MsgBox % LOBYTE(var)
```

# LOWORD

Retrieves the low-order word from the given value.

```
OutputVar := LOWORD(Value)
```

```
Function Example: Word := LOWORD(1310730)
```

## Parameters

### OutputVar

The name of the variable to store the low-order word.

### Value

The value to convert to low-order word.

## Examples

```
var:=1310730
MsgBox % LOWORD(var)
```

# MAKELANGID

Creates a [language identifier](#) from a primary language identifier and a sublanguage identifier.

```
OutputVar := MAKELANGID(PrimaryLanguage, SubLanguage)
```

```
Function Example: langid := MAKELANGID(0x9,
0x1)
```

## Parameters

### OutputVar

The name of the variable to store the language identifier.

### PrimaryLanguage

The primary language identifier.

### SubLanguage

The sublanguage identifier.

## Examples

```
MsgBox % MakeLangId(0x9, 0x1) ; English, English
```

# MAKELCID

Creates a [locale identifier](#) from a [language identifier](#) and a [sort order identifier](#).

```
OutputVar := MAKELCID(LanguageID, SortID)
```

```
Function Example: langid := MAKELCID(0x9, 0x1)
```

## Parameters

### OutputVar

The name of the variable to store the language identifier.

### LanguageID

The language identifier.

### SortID

Sort order identifier.

## Examples

```
MsgBox % MakeLCId(MakeLangID(0x9, 0x1), 0) ;
English, English, SORT_DEFAULT
```

# MAKELONG

Creates a LONG value by concatenating the specified values.

```
OutputVar := MAKELONG(ValueLow, ValueHigh)
```

```
Function Example: Long := MAKELONG(10, 20)
```

## Parameters

### OutputVar

The name of the variable to store the LONG value.

### ValueLow

The low-order value.

### ValueHigh

The high-order value.

## Examples

```
MsgBox % MakeLong(10, 20)
```

# MAKELPARAM

Creates a value for use as an lParam parameter in a message.

```
OutputVar := MAKELPARAM(ValueLow, ValueHigh)
```

```
Function Example: lParam := MAKELPARAM(10, 20)
```

## Parameters

### OutputVar

The name of the variable to store the lParam value.

### ValueLow

The low-order value.

### ValueHigh

The high-order value.

## Examples

```
MsgBox % MakeLParam(10, 20)
```

# MAKELRESULT

Creates a value for use as a return value from a window procedure.

```
OutputVar := MAKELRESULT(ValueLow, ValueHigh)
```

```
Function Example: lResult := MAKELRESULT(10,
20)
```

## Parameters

### OutputVar

The name of the variable to store the lResult value.

### ValueLow

The low-order value.

### ValueHigh

The high-order value.

## Examples

```
MsgBox % MakeLResult(10, 20)
```

# MAKEWORD

Creates a value for use as a return value from a window procedure.

```
OutputVar := MAKEWORD(ValueLow, ValueHigh)
```

```
Function Example: word := MAKEWORD(10, 20)
```

## Parameters

### OutputVar

The name of the variable to store the WORD value.

### ValueLow

The low-order value.

### ValueHigh

The high-order value.

## Examples

```
MsgBox % MakeWord(10, 20)
```

# MAKEWPARAM

Creates a value for use as an wParam parameter in a message.

```
OutputVar := MAKEWPARAM(ValueLow, ValueHigh)
```

```
Function Example: wParam := MAKEWPARAM(10, 20)
```

## Parameters

### OutputVar

The name of the variable to store the lParam value.

### ValueLow

The low-order value.

### ValueHigh

The high-order value.

## Examples

```
MsgBox % MakeWParam(10, 20)
```

# MCodeH

Creates a [DynaCall](#) object for machine code function.

```
OutputVar := MCodeH(Hex [, Definition, Parameter1,
Parameter2, ...])
```

```
Function Example: BSwap16 :=
MCodeH("8AE18AC5C3", "h==h")
```

## Parameters

### OutputVar

The name of the variable in which to store the function object.

### Definition

Same definition syntax as used in [DynaCall](#).

### Parameter1, Parameter2, ... (optional)

Default Parameters to use when calling the function, same as for [DynaCall](#).

## Related

[DllCall](#), [#DllImport](#), [DynaCall](#), [WinApi](#)

## Examples

```
RGB_TO_BGR:=MCodeH(A_PtrSize=4?"8B4C24040FB6C18BD1
C1E01081E200FF00000BC2C1E9100BC1C3":"0FB6C18BD1C1E
910C1E01081E200FF00000BC20BC1C3","ui==ui")
MsgBox % format("0x{1:X}",RGB_TO_BGR[0xAABBCC])
```

# ResDelete

Deletes a resource in executable file (dll or exe).

```
OutputVar := ResDelete(Executable, Name [, Type, Language])
```

```
Command Example: ResDelete A_AhkPath,
"MYRESOURCE"
```

```
Function Example: success :=
ResDelete(A_AhkPath, "MYRESOURCE")
```

## Parameters

### OutputVar

The name of the variable in which to store true / 1 if resource was deleted or does not exist or false / 0 otherwise.

### Executable

Path to the executable file (dll or exe).

### Name

The name of resource.

### Type (optional)

Type of resource. Default is RCDATA (10).  
See [MSDN](#) for default resource types.

## Language (optional)

Resource language, default 1033 (en-us).

## Related

[ResPut](#), [ResGet](#), [ResPutFile](#), [ResExist](#), [ResDelete](#), [ResDllCreate](#),  
[UnZipRawMemory](#)

## Example

```
ResDelete(A_AhkPath, "MYRESOURCE")
```

# ResDllCreate

Creates a resource only dll. Such dll will not have any executable code but can be loaded into programs to access resources.

```
OutputVar := ResDllCreate(DllPath)
```

```
Command Example: ResDllCreate A_ScriptDir
"\MyDll.dll"
```

```
Function Example: ResDllCreate(A_ScriptDir
"\MyDll.dll")
```

## Parameters

### OutputVar

The name of the variable to store true / 1 if the dll was written successfully or 0 / false otherwise.

### DllPath

Path for the dll to be created.

## ErrorLevel

ErrorLevel will be set to 1 (true) if the file could not be created, otherwise it will be 0 (false).

## Related

ResGet, ResPut, ResPutFile, ResExist, ResDelete, ResDllCreate,  
UnZipRawMemory

## Example

```
ResDllCreate, success, %A_ScriptDir%\MyDll.dll
If !success
 MsgBox The dll could not be created.
```

# ResExist

Check whether resource exists in the executable file (dll or exe).

```
OutputVar := ResExist(Executable, Name [, Type, Language])
```

```
Function Example: Exist := ResExist(A_AhkPath, "MYRESOURCE")
```

## Parameters

### OutputVar

The name of the variable in which to store true / 1 if the resource exist or false / 0 otherwise.

### Executable

Path to the executable file (dll or exe).

### Name

The name of resource.

### Type (optional)

Type of resource. Default is RCDATA (10).  
See [MSDN](#) for default resource types.

### Language (optional)

Language of resource. Default is 1033.

## Related

[ResGet](#), [ResPut](#), [ResPutFile](#), [ResExist](#), [ResDelete](#), [ResDllCreate](#),  
[UnZipRawMemory](#)

## Example

```
If !ResExist(A_AhkPath,"RESEXIST.AHK","LIB")
 MsgBox ResExist.ahk is not included in resource
 library.
```

# ResPut

Updates a resource in executable file (dll or exe).

```
OutputVar := ResPut(Data, Size, Executable, Name, Type, Language)
```

```
Command Example: ResPut
Data, VarsetCapacity(data), A_ScriptDir
"\MyDll.dll", "MYRESOURCE"
```

```
Function Example: Success := ResPut(Data,
VarsetCapacity(data), A_ScriptDir "\MyDll.dll",
"MYRESOURCE")
```

## Parameters

### OutputVar

The name of the variable in which to store true / 1 if resource was written and false / 0 otherwise.

### Data (ByRef)

[ByRef variable](#) or a pointer to data to be written.

### Size

Size of data to be written in bytes.

### Executable

Path to the executable file (dll or exe).

## Name

The name of resource.

## Type (optional)

Type of resource. Default is RCDATA (10).

See [MSDN](#) for default resource types.

## Language (optional)

The language of resource, default 1033.

## Remarks

If you wish to compress or encrypt your data you can use [ZipFileRaw](#).

## Related

[ResGet](#), [ResPutFile](#), [ResExist](#), [ResDelete](#), [ResDllCreate](#), [UnZipRawMemory](#)

## Example

```
FileRead,Data,*c %A_ScriptDir%\MYRES.RES
FileGetSize,sz,%A_ScriptDir%\MYRES.RES
ResPut(Data, sz, A_AhkPath, "MYRES.RES")
```

# ResPutFile

Updates a resource in executable file (dll or exe).

```
OutputVar := ResPutFile(FilePath, Executable, Name [,
Type, Language])
```

```
Command Example: ResPut A_ScriptDir
"\MyFile.txt", A_ScriptDir "\MyDll.dll",
"MYRESOURCE"
```

```
Function Example: Success := ResPut(A_ScriptDir
"\MyFile.txt", A_ScriptDir "\MyDll.dll",
"MYRESOURCE")
```

## Parameters

### OutputVar

The name of the variable in which to store true / 1 if resource was written and false / 0 otherwise.

### FilePath

File to be included in resources.

### Executable

Path to the executable file (dll or exe).

### Name

The name of resource.

### Type (optional)

Type of resource. Default is RCDATA (10).  
See [MSDN](#) for default resource types.

### Language (optional)

The language of resource, default 1033.

## Remarks

If you wish to compress or encrypt your data you can use [ZipFileRaw](#).

## Related

[ResGet](#), [ResPutFile](#), [ResExist](#), [ResDelete](#), [ResDllCreate](#), [UnZipRawMemory](#)

## Example

```
FileRead,Data,*c %A_ScriptDir%\MYRES.RES
FileGetSize,sz,%A_ScriptDir%\MYRES.RES
ResPut(Data, sz, A_AhkPath, "MYRES.RES")
```

# StrPut

Copies a string to a variable, optionally converting to or from a given code page.

```
OutputVar := StrPutVar(String, Variable [, Encoding])
```

```
Command Example: StrPutVar "AutoHotkey", Var,
"CP0"
```

```
Function Example: size :=
StrPutVar("AutoHotkey", Var, "CP0")
```

## Parameters

### OutputVar

The name of the variable in which to store size of new string in bytes including terminator.

### String

Any string. Numbers are also acceptable.

### Variable (ByRef)

Output variable to store new string.

### Encoding (optional)

The source encoding for StrGet or target encoding for StrPut; for example, "UTF-8", "UTF-16" or "CP936". Specify "CP0" to use the system default ANSI code page. When omitted UTF-8 is used.

## Remarks

If conversion between code pages is necessary, the required buffer size may differ from the size of the source *String*.

## Related

[Script Compatibility](#), [StrGet](#), [StrPut](#), [FileEncoding](#), [VarSetCapacity](#)

## Examples

```
MsgBox % StrPutVar("AutoHotkey",var,"CP0")
```

# ToChar

Convert an integer to signed char type.

```
OutputVar := ToChar(ByRef IntOrVar)
```

```
Command Example: ToChar var
Function Example: chr := ToChar(255)
```

## Parameters

### OutputVar

The name of the variable in which to store the converted integer.

### ByRef IntOrVar

The value to convert, if a variable is passed, converted value will be written to it.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#), [WinApi](#)

# ToInt

Convert an integer to signed int type.

```
OutputVar := ToInt(ByRef IntOrVar)
```

```
Command Example: ToInt var
Function Example: int := ToInt(4294967295)
```

## Parameters

### OutputVar

The name of the variable in which to store the converted integer.

### ByRef IntOrVar

The value to convert, if a variable is passed, converted value will be written to it.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#), [WinApi](#)

# ToShort

Convert an integer to signed short type.

```
OutputVar := ToShort(ByRef IntOrVar)
```

```
Command Example: ToShort var
Function Example: short := ToShort(65535)
```

## Parameters

### OutputVar

The name of the variable in which to store the converted integer.

### ByRef IntOrVar

The value to convert, if a variable is passed, converted value will be written to it.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#), [WinApi](#)

# ToUChar

Convert an integer to unsigned char type.

```
OutputVar := ToUChar(ByRef IntOrVar)
```

```
Command Example: ToUChar var
Function Example: uchr := ToUChar(-1)
```

## Parameters

### OutputVar

The name of the variable in which to store the converted integer.

### ByRef IntOrVar

The value to convert, if a variable is passed, converted value will be written to it.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#), [WinApi](#)

# ToUInt

Convert an integer to unsigned int type.

```
OutputVar := ToUInt(ByRef IntOrVar)
```

```
Command Example: ToUInt var
Function Example: uint := ToUInt(-1)
```

## Parameters

### OutputVar

The name of the variable in which to store the converted integer.

### ByRef IntOrVar

The value to convert, if a variable is passed, converted value will be written to it.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#), [WinApi](#)

# ToUShort

Convert an integer to unsigned short type.

```
OutputVar := ToUShort(ByRef IntOrVar)
```

```
Command Example: ToUShort var
Function Example: ushort := ToUShort(-1)
```

## Parameters

### OutputVar

The name of the variable in which to store the converted integer.

### ByRef IntOrVar

The value to convert, if a variable is passed, converted value will be written to it.

## Related

[DllCall](#), [DynaCall](#), [#DllImport](#), [WinApi](#)

# CriticalObject

Built-in function to enwrap an object for multi-thread use. Such objects can be used from multiple threads without causing a crash.

```
OutputVar := CriticalObject(Object,
lpCriticalSection)
```

```
Function Example: obj :=
CriticalObject(MyCriticalSection)
```

## Parameters

### OutputVar

The name of the variable in which to store the created object.

### Object

Existing Object or CriticalObject to use, this can be also a pointer.

When CriticalObject is given, its [CriticalSection](#) will be used and second parameter will be ignored.

When this parameter is empty new object will be created and used.

### lpCriticalSection

Pointer to a [CriticalSection](#) to use. When this parameter is omitted new CriticalSection will be created unless CriticalObject was given in first parameter, then its CriticalSection will be used.

## General Remarks

### How does it work:

- First the CriticalSection is locked so the object cannot be used in other threads
- Then the referenced object is invoked
- Afterwards critical section is unlocked so other threads can use the object again

To retrieve the original object from CriticalObject use:

```
object := CriticalObject(CriticalObject,1)
```

To retrieve the pointer to CriticalSection use:

```
lpCriticalSection :=
CriticalObject(CriticalObject,2)
```

When last reference to internal object is deleted, CriticalSection is deleted as well.

## Related

[Object](#)

## Examples

```
obj := CriticalObject() ; Create new critical
object
Loop, 4 ; Create 4 Threads.
 AhkThread%A_Index% :=
 AhkThread("obj:=CriticalObject(" (&obj;)
 ")`nLoop`nobj[" A_Index "]:= A_Index")

Loop ; Show current content of object.
 ToolTip, % obj.1 "`n" obj.2 "`n" obj.3 "`n" obj.4
Esc::ExitApp
```

# ObjByRef

Creates an object containing `ByRef` variables, `obj.var` returns actual variable content and `obj.var := value` will assign new value to variable.

```
OutputVar := ObjByRef(Var1 [, Var2, ...])
```

```
Function Example: ByRefObj := ObjByRef(Var1)
```

## Parameters

### OutputVar

The name of the variable in which to store the object.

### Var1, Var2, ...

The name of the variable which reference will be stored in the object, key will be the name of the variable.

## General Remarks

`ByRefObj` can also accept a new variable, simply use `ByRefObj.newvar:=newvar`. Other than calling `ObjByRef()` where `key = VarName`, when assigning new variable any name for the key can be used, e.g. `ByRefObj.MyName:=Name`

## Examples

```
ByRefObj := ObjByRef(var:="Hello World!")
MsgBox % ByRefObj.var ; returns content of var
ByRefObj.var := "Hello AHK!" ; assign new content
to var
MsgBox % var
ByRefObj.newvar:=var2 ; add new variable to
ByRefObj but use newvar instead of var2 as key
name
ByRefObj.newvar:=1 ; assign new content to var
MsgBox % var2
```

# ObjDump

Dump an object to memory or save to file for later use.

```
OutputVar := ObjDump(ObjOrPath [, DumpToVar, Mode,
Password])
```

```
Command Example: ObjDump
"C:\Temp\MyObject.dmp", obj
```

```
Function Example: size := ObjDump(obj, var)
```

## Parameters

### OutputVar

The name of the variable in which to store the size of dumped object.

### ObjOrPath

The object to be dumped to memory.

To save the object to a file this parameter must be the file path and DumpToVar parameter the object, see Examples.

### DumpToVar (optional)

The name of the variable in which to store the dumped memory. When this parameter is omitted only the size will be returned.

When saving the object to a file this parameter will be the object and ObjOrPath the filepath.

### Mode (optional)

This parameter has effect on values of type string only and determines whether the dumped object will be zipped.

Supported modes are 0 (default), 1 (dump complete memory), 2 (default + compression), 3 (dump complete memory and compress).

When Mode is 2 or 3, dumped object will be compressed and you can use a password to encrypt the data.

When Mode is 1 or 3, complete memory of string values is saved using `obj.GetCapacity(key)`. Otherwise only the actual string including terminating character is saved.

### Password (optional)

When Mode parameter is omitted or 0 or 1, this parameter can be a passwords to use for dumped object.

## General Remarks

Only standard objects and arrays are supported, `ComObject`, `RegexMatchObject`, `FileObject`, `Func` and `DynaCall` objects are not supported.

`Struct` object can be dumped but `ObjLoad` will restore a normal object instead of a `Struct` object.

The "base" reference is not saved in `ObjLoad` and is not restored.

By default (Mode = 0) and Mode = 1 the dumped object is compressed and a password can be used to encrypt the dumped object.

## Related

[ObjLoad](#)

## Examples

```
; Create a simple object
obj := {key:"value",1:"test",2:10}

; Dump an object to variable.
sz := ObjDump(obj, var)

; Restore an object from variable
obj := ObjLoad(&var;)

; Dump an object to file
sz := ObjDump(A_ScriptDir "\MyDump.bin", obj)

; Restore an object from file
obj := ObjLoad(A_ScriptDir "\MyDump.bin")

; Get the dump size of an object
MsgBox % ObjDump(obj)
```

# ObjLoad

Load a [dumped](#) object from memory or file.

```
OutputVar := ObjLoad(AddressOrPath [, Size, Password])
```

```
Function Example: obj := ObjLoad(A_ScriptDir
"\MyObj.bin")
```

## Parameters

### OutputVar

The name of the variable in which to store the loaded object.

### AddressOrPath

The address of a variable or a pointer to dumped object in memory.  
This can be also a file path to load an object from file.

### Size (optional)

When AddressOrPath is an address and [ObjDump](#) used compression (mode 2 or 3) this must be the size of the dumped object in memory, otherwise it can be omitted.

### Password (optional)

The password if for dumped object if it was used in [ObjDump](#), otherwise it can be omitted.

## General Remarks

Only standard objects and arrays are supported (such that can be enumerated by [for loop](#)).

ComObject, RegExMatchObject, FileObject, Func and DynaCall objects are not supported.

[Struct](#) object can be [dumped](#) but ObjLoad will restore a normal object instead of a Struct object.

If original object had a "base" reference, it was not saved by [ObjDump](#) and is not restored.

## Examples

See [ObjDump](#).

# ObjShare

Included function to use an object in a multi-threaded environment. This is especially useful to call methods of a class from a different thread. Such objects can be used from multiple threads without causing a crash using COM functions `LresultFromObject` and `ObjectFromLresult`.

```
OutputVar := ObjShare(ObjectOrLresult)
```

```
Function Example: Lresult := ObjShare(MyObject)
```

## Parameters

### OutputVar

The name of the variable in which to store the Lresult created for object or IDispatch proxy COM object.

### ObjectOrLresult

Existing Object or Lresult to use.

When this parameter is an object OutputVar will be Lresult value.

Otherwise when Lresult is passed OutputVar will be an IDispatch proxy COM object

## General Remarks

AutoHotkey is not thread safe and when you execute code from another

thread while main thread is executing code as well it will result in an access violation and your program will crash.

### How does ObjShare work:

- Lresult is created for an object in any exe or dll thread that owns the object.
- This Lresult is used in a different thread to create an IDispatch proxy COM object that when called will invoke the original object.
- While IDispatch proxy COM object is processed, the thread where lresult was created will be able to execute any other code before finishing processing.

Note! You cannot access the object in the other thread using [ahkExec](#), [ahkFunction](#) or [ahkLabel](#) (GoSub mode), this will result in a COM error.

When the thread that created lresult is in critical mode you will not be able to invoke the IDispatch proxy COM object until thread becomes non critical.

To share an object to multiple threads you will need to create a separate lresult for each thread by calling ObjShare multiple times.

## Related

[Object](#), [CriticalSection](#)

## Examples

```

Gui,Add,Slider,vSlider,50
Gui,Add,CheckBox
Gui,Show,w200 h100
t:=new Test
lresult := ObjShare(t) ; Create lresult to use in
a different thread
AhkThread("
(
 t := ObjShare(%lresult%) ; Create IDispatch
proxy COM object
 Loop
 t.CheckBox()
)")

Loop
 t.Slider()
return
GuiClose:
ExitApp
Class Test {
 CheckBox(){
 GuiControlGet,value,,Button1
 GuiControl,,Button1,% Value?0:1
 }
 Slider(){
 GuiControlGet,value,,Slider
 if value=100
 value:=0
 GuiControl,,Slider,% Value+1
 }
}

```

# Mouse Gestures -- by deguix

This script watches how you move the mouse whenever the right mouse button is being held down. If it sees you "draw" a recognized shape or symbol, it will launch a program or perform another custom action of your choice (just like hotkeys). See the included README file for how to define gestures.

[Download This Script \(17 KB Zip File\)](#) | [Other Sample Scripts](#) | [Home](#)

# Easy Window Dragging (requires XP/2k/NT)

Normally, a window can only be dragged by clicking on its title bar. This script extends that so that any point inside a window can be dragged. To activate this mode, hold down CapsLock or the middle mouse button while clicking, then drag the window to a new position.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; Note: You can optionally release Capslock or the
middle mouse button after
; pressing down the mouse button rather than
holding it down the whole time.
```

```
~MButton & LButton::
CapsLock & LButton::
CoordMode, Mouse ; Switch to screen/absolute
coordinates.
MouseGetPos, EWD_MouseStartX, EWD_MouseStartY,
EWD_MouseWin
WinGetPos, EWD_OriginalPosX, EWD_OriginalPosY,,
ahk_id %EWD_MouseWin%
WinGetMinMax, EWD_WinState, ahk_id %EWD_MouseWin%
if !EWD_WinState ; Only if the window isn't
maximized
SetTimer, EWD_WatchMouse, 10 ; Track the mouse
as the user drags it.
return

EWD_WatchMouse:
GetKeyState, EWD_LButtonState, LButton, P
```

```

if !EWD_LButtonState ; Button has been released,
so drag is complete.
{
 SetTimer,, off
 return
}
GetKeyState, EWD_EscapeState, Escape, P
if EWD_EscapeState ; Escape has been pressed, so
drag is cancelled.
{
 SetTimer,, off
 WinMove, ahk_id %EWD_MouseWin%,,
 %EWD_OriginalPosX%, %EWD_OriginalPosY%
 return
}
; Otherwise, reposition the window to match the
change in mouse coordinates
; caused by the user having dragged the mouse:
CoordMode, Mouse
MouseGetPos, EWD_MouseX, EWD_MouseY
WinGetPos, EWD_WinX, EWD_WinY,, ahk_id
%EWD_MouseWin%
SetWinDelay, -1 ; Makes the below move
faster/smoother.
WinMove, ahk_id %EWD_MouseWin%,, % EWD_WinX +
EWD_MouseX - EWD_MouseStartX, % EWD_WinY +
EWD_MouseY - EWD_MouseStartY
EWD_MouseStartX := EWD_MouseX ; Update for the
next timer-call to this subroutine.
EWD_MouseStartY := EWD_MouseY
return

```

# Easy Window Dragging -- KDE style (requires XP/2k/NT) -- by Jonny

This script makes it much easier to move or resize a window: 1) Hold down the ALT key and LEFT-click anywhere inside a window to drag it to a new location; 2) Hold down ALT and RIGHT-click-drag anywhere inside a window to easily resize it; 3) Press ALT twice, but before releasing it the second time, left-click to minimize the window under the mouse cursor, right-click to maximize it, or middle-click to close it.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; This script was inspired by and built on many
like it
; in the forum. Thanks go out to ck, thinkstorm,
Chris,
; and aurelian for a job well done.

; Change history:
; July 16, 2016: Revised code for AHK v2
compatibility
; November 07, 2006: Optimized resizing code in
!RButton, courtesy of bluedawn.
; February 05, 2006: Fixed double-alt (the ~Alt
hotkey) to work with latest versions of AHK.

; The Double-Alt modifier is activated by pressing
; Alt twice, much like a double-click. Hold the
second
; press down until you click.
;
```

```

; The shortcuts:
; Alt + Left Button : Drag to move a window.
; Alt + Right Button : Drag to resize a window.
; Double-Alt + Left Button : Minimize a window.
; Double-Alt + Right Button : Maximize/Restore a
window.
; Double-Alt + Middle Button : Close a window.
;
; You can optionally release Alt after the first
; click rather than holding it down the whole
time.

; This is the setting that runs smoothest on my
; system. Depending on your video card and cpu
; power, you may want to raise or lower this
value.
SetWinDelay,2

CoordMode,Mouse
return

!LButton::
If DoubleAlt
{
 MouseGetPos,,,KDE_id
 ; This message is mostly equivalent to
WinMinimize,
 ; but it avoids a bug with PSPad.
 PostMessage,0x112,0xf020,,,ahk_id %KDE_id%
 DoubleAlt := false
 return
}
; Get the initial mouse position and window id,
and
; abort if the window is maximized.
MouseGetPos,KDE_X1,KDE_Y1,KDE_id
WinGetMinMax,KDE_Win,ahk_id %KDE_id%

```

```

If KDE_Win
 return
; Get the initial window position.
WinGetPos,KDE_WinX1,KDE_WinY1,,ahk_id %KDE_id%
Loop
{
 GetKeyState,KDE_Button,LButton,P ; Break if
button has been released.
 If !KDE_Button
 break
 MouseGetPos,KDE_X2,KDE_Y2 ; Get the current
mouse position.
 KDE_X2 -= KDE_X1 ; Obtain an offset from the
initial mouse position.
 KDE_Y2 -= KDE_Y1
 KDE_WinX2 := (KDE_WinX1 + KDE_X2) ; Apply this
offset to the window position.
 KDE_WinY2 := (KDE_WinY1 + KDE_Y2)
 WinMove,ahk_id
%KDE_id%,,%KDE_WinX2%,%KDE_WinY2% ; Move the
window to the new position.
}
return

!RButton::
If DoubleAlt
{
 MouseGetPos,,,KDE_id
; Toggle between maximized and restored state.
WinGetMinMax,KDE_Win,ahk_id %KDE_id%
If KDE_Win
 WinRestore,ahk_id %KDE_id%
Else
 WinMaximize,ahk_id %KDE_id%
DoubleAlt := false
return
}

```

```

; Get the initial mouse position and window id,
and
; abort if the window is maximized.
MouseGetPos,KDE_X1,KDE_Y1,KDE_id
WinGetMinMax,KDE_Win,ahk_id %KDE_id%
If KDE_Win
 return
; Get the initial window position and size.
WinGetPos,KDE_WinX1,KDE_WinY1,KDE_WinW,KDE_WinH,ahk_id %KDE_id%
; Define the window region the mouse is currently
in.
; The four regions are Up and Left, Up and Right,
Down and Left, Down and Right.
If (KDE_X1 < KDE_WinX1 + KDE_WinW / 2)
 KDE_WinLeft := 1
Else
 KDE_WinLeft := -1
If (KDE_Y1 < KDE_WinY1 + KDE_WinH / 2)
 KDE_WinUp := 1
Else
 KDE_WinUp := -1
Loop
{
 GetKeyState,KDE_Button,RButton,P ; Break if
button has been released.
 If !KDE_Button
 break
 MouseGetPos,KDE_X2,KDE_Y2 ; Get the current
mouse position.
 ; Get the current window position and size.

WinGetPos,KDE_WinX1,KDE_WinY1,KDE_WinW,KDE_WinH,ahk_id %KDE_id%
 KDE_X2 -= KDE_X1 ; Obtain an offset from the
initial mouse position.
 KDE_Y2 -= KDE_Y1

```

```

; Then, act according to the defined region.
WinMove,ahk_id %KDE_id%,, % KDE_WinX1 +
(KDE_WinLeft+1)/2*KDE_X2 ; X of resized window
, % KDE_WinY1 +
(KDE_WinUp+1)/2*KDE_Y2 ; Y of resized window
, % KDE_WinW -
KDE_WinLeft *KDE_X2 ; W of resized window
, % KDE_WinH -
KDE_WinUp *KDE_Y2 ; H of resized window
KDE_X1 := (KDE_X2 + KDE_X1) ; Reset the
initial position for the next iteration.
KDE_Y1 := (KDE_Y2 + KDE_Y1)
}
return

```

```

; "Alt + MButton" may be simpler, but I
; like an extra measure of security for
; an operation like this.

```

```

!MButton::
If DoubleAlt
{
 MouseGetPos,,,KDE_id
 WinClose,ahk_id %KDE_id%
 DoubleAlt := false
 return
}
return

```

```

; This detects "double-clicks" of the alt key.
~Alt::
DoubleAlt := A_PriorHotKey = "~Alt" AND
A_TimeSincePriorHotkey < 400
Sleep 0
KeyWait Alt ; This prevents the keyboard's auto-
repeat feature from interfering.
return

```

# Easy Access to Favorite Folders -- by Savage

When you click the middle mouse button while certain types of windows are active, this script displays a menu of your favorite folders. Upon selecting a favorite, the script will instantly switch to that folder within the active window. The following window types are supported: 1) Standard file-open or file-save dialogs; 2) Explorer windows; 3) Console (command prompt) windows. The menu can also be optionally shown for unsupported window types, in which case the chosen favorite will be opened as a new Explorer window.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; CONFIG: CHOOSE YOUR HOTKEY
; If your mouse has more than 3 buttons, you could
try using
; XButton1 (the 4th) or XButton2 (the 5th) instead
of MButton.
; You could also use a modified mouse button (such
as ^MButton) or
; a keyboard hotkey. In the case of MButton, the
tilde (~) prefix
; is used so that MButton's normal functionality
is not lost when
; you click in other window types, such as a
browser. The presence
; of a tilde tells the script to avoid showing the
menu for
; unsupported window types. In other words, if
there is no tilde,
; the hotkey will always display the menu; and
```

```
upon selecting a
; favorite while an unsupported window type is
active, a new
; Explorer window will be opened to display the
contents of that
; folder.
```

```
f_Hotkey := "~MButton"
```

```
; CONFIG: CHOOSE YOUR FAVORITES
; Update the special commented section below to
list your favorite
; folders. Specify the name of the menu item
first, followed by a
; semicolon, followed by the name of the actual
path of the favorite.
; Use a blank line to create a separator line.
```

```
/*
ITEMS IN FAVORITES MENU <-- Do not change this
string.
```

```
Desktop ; %A_Desktop%
Favorites ; %A_Desktop%\..\Favorites
My Documents ; %A_MyDocuments%
```

```
Program Files; %A_ProgramFiles%
```

```
*/
```

```
; END OF CONFIGURATION SECTION
; Do not make changes below this point unless you
want to change
; the basic functionality of the script.
```

```
#SingleInstance ; Needed since the hotkey is
dynamically created.
```

```
Hotkey, %f_Hotkey%, f_DisplayMenu
```

```

f_HotkeyFirstChar := SubStr(f_Hotkey, 1, 1)
if f_HotkeyFirstChar = "~" ; Show menu only for
certain window types.
 f_AlwaysShowMenu := false
else
 f_AlwaysShowMenu := true

; Used to reliably determine whether script is
compiled:
SplitPath, %A_ScriptName%,,, f_FileExt
if f_FileExt = "Exe" ; Read the menu items from
an external file.
 f_FavoritesFile :=
"%A_ScriptDir%\Favorites.ini"
else ; Read the menu items directly from this
script file.
 f_FavoritesFile := A_ScriptFullPath

;----Read the configuration file.
f_AtStartingPos := false
f_MenuItemCount := 0
Loop, Read, %f_FavoritesFile%
{
 if f_FileExt <> "Exe"
 {
 ; Since the menu items are being read
directly from this
 ; script, skip over all lines until the
starting line is
 ; arrived at.
 if !f_AtStartingPos
 {
 if InStr(A_LoopReadLine, "ITEMS IN
FAVORITES MENU")
 f_AtStartingPos := true
 continue ; Start a new loop
iteration.
 }
 }
}

```

```

 }
 ; Otherwise, the closing comment symbol
marks the end of the list.
 if A_LoopReadLine = "*/"
 break ; terminate the loop
 }
 ; Menu separator lines must also be counted to
be compatible
 ; with A_ThisMenuItemPos:
 f_MenuItemCount++
 if !A_LoopReadLine ; Blank indicates a
separator line.
 Menu, Favorites, Add
 else
 {
 f_line := StrSplit(A_LoopReadLine, ";", "
`t")
 ; Resolve any references to variables
within either field, and
 ; create a new array element containing
the path of this favorite:
 f_path%f_MenuItemCount% :=
Deref(f_line[2])
 f_line[1] := Deref(f_line[1])
 Menu, Favorites, Add, % f_line[1],
f_OpenFavorite
 }
}
return ;----End of auto-execute section.

;----Open the selected favorite
f_OpenFavorite:
; Fetch the array element that corresponds to the
selected menu item:
f_path := f_path%A_ThisMenuItemPos%
if f_path = ""

```

```

return
if f_class = "#32770" ; It's a dialog.
{
 ; Activate the window so that if the user is
middle-clicking
 ; outside the dialog, subsequent clicks will
also work:
WinActivate ahk_id %f_window_id%
 ; Retrieve any filename that might already be
in the field so
 ; that it can be restored after the switch to
the new folder:
ControlGetText, f_text, Edit1, ahk_id
%f_window_id%
ControlSetText, Edit1, %f_path%, ahk_id
%f_window_id%
ControlFocus, Edit1, ahk_id %f_window_id%
ControlSend, Edit1, {Enter}, ahk_id
%f_window_id%
Sleep, 100 ; It needs extra time on some
dialogs or in some cases.
ControlSetText, Edit1, %f_text%, ahk_id
%f_window_id%
return
 ; else fall through to the bottom of the
subroutine to take standard action.
}
else if f_class ~= "ExploreWClass|CabinetWClass"
; In Explorer, switch folders.
{
ControlClick, ToolbarWindow323, ahk_id
%f_window_id%,,,, NA x1 y1
ControlSetText, Edit1, %f_path%, ahk_id
%f_window_id%
 ; Tekl reported the following: "If I want to
change to Folder L:\folder
 ; then the addressbar shows

```

```

http://www.L:\folder.com. To solve this,
; I added a {right} before {Enter}":
 ControlSend, Edit1, {Right}{Enter}, ahk_id
%f_window_id%
 return
; else fall through to the bottom of the
subroutine to take standard action.
}
else if f_class = "ConsoleWindowClass" ; In a
console window, CD to that directory
{
 WinActivate, ahk_id %f_window_id% ; Because
sometimes the mclick deactivates it.
 SetKeyDelay, 0 ; This will be in effect only
for the duration of this thread.
 if InStr(f_path, ":") ; It contains a drive
letter
 {
 f_path_drive := SubStr(f_path, 1, 1)
 Send %f_path_drive%:{enter}
 }
 Send, cd %f_path%{Enter}
 return
}
; Since the above didn't return, one of the
following is true:
; 1) It's an unsupported window type but
f_AlwaysShowMenu is y (yes).
Run, Explorer %f_path% ; Might work on more
systems without double quotes.
return

```

```

;----Display the menu

```

```

f_DisplayMenu:

```

```

; These first few variables are set here and used
by f_OpenFavorite:

```

```
WinGetID, f_window_id, A
WinGetClass, f_class, ahk_id %f_window_id%
if f_AlwaysShowMenu = false ; The menu should be
shown only selectively.
{
 if !(f_class ~=
"#32770|ExploreWClass|CabinetWClass|ConsoleWindowC
lass")
 return ; Since it's some other window
type, don't display menu.
}
; Otherwise, the menu should be presented for this
type of window:
Menu, Favorites, show
return
```

# IntelliSense -- by Rajat (requires XP/2k/NT)

This script watches while you edit an AutoHotkey script. When it sees you type a command followed by a comma or space, it displays that command's parameter list to guide you. In addition, you can press Ctrl+F1 (or another hotkey of your choice) to display that command's page in the help file. To dismiss the parameter list, press Escape or Enter.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; CONFIGURATION SECTION: Customize the script with
the following variables.
```

```
; The hotkey below is pressed to display the
current command's page in the
```

```
; help file:
```

```
I_HelpHotkey := "^F1"
```

```
; The string below must exist somewhere in the
active window's title for
```

```
; IntelliSense to be in effect while you're
typing. Make it blank to have
```

```
; IntelliSense operate in all windows. Make it
Pad to have it operate in
```

```
; editors such as Metapad, Notepad, and Textpad.
Make it .ahk to have it
```

```
; operate only when a .ahk file is open in
Notepad, Metapad, etc.
```

```
I_Editor := "pad"
```

```
; If you wish to have a different icon for this
```

```
script to distinguish it from
; other scripts in the tray, provide the filename
below (leave blank to have
; no icon). For example:
```

```
E:\stuff\Pics\icons\GeoIcons\Information.ico
```

```
I_Icon := ""
```

```
; END OF CONFIGURATION SECTION (do not make
changes below this point unless
; you want to change the basic functionality of
the script).
```

```
SetKeyDelay, 0
#SingleInstance
```

```
if I_HelpHotkey <> ""
 Hotkey, %I_HelpHotkey%, I_HelpHotkey
```

```
; Change tray icon (if one was specified in the
configuration section above):
```

```
if I_Icon <> ""
 if FileExist(I_Icon)
 Menu, Tray, Icon, %I_Icon%
```

```
; Determine AutoHotkey's location:
```

```
RegRead, ahk_dir,
HKEY_LOCAL_MACHINE\SOFTWARE\AutoHotkey, InstallDir
if ErrorLevel ; Not found, so look for it in some
other common locations.
```

```
{
 if A_AhkPath
 SplitPath, %A_AhkPath%,, ahk_dir
 else if FileExist("../..\AutoHotkey.chm")
 ahk_dir := "../.."
 else if
FileExist("%A_ProgramFiles%\AutoHotkey\AutoHotkey.
chm")
```

```

 ahk_dir := "%A_ProgramFiles%\AutoHotkey"
else
{
 MsgBox Could not find the AutoHotkey
folder.
 ExitApp
}
}

ahk_help_file := "%ahk_dir%\AutoHotkey.chm"

; Read command syntaxes; can be found in AHK
Basic, but it's outdated:
Loop, Read,
%ahk_dir%\Extras\Editors\Syntax\Commands.txt
{
 I_FullCmd := A_LoopReadLine

 ; Directives have a first space instead of a
first comma.
 ; So use whichever comes first as the end of
the command name:
 I_cPos := InStr(I_FullCmd, ",")
 I_sPos := InStr(I_FullCmd, " ")
 if (!I_cPos or (I_cPos > I_sPos and I_sPos))
 I_EndPos := I_sPos
 else
 I_EndPos := I_cPos

 if I_EndPos
 I_CurrCmd := SubStr(I_FullCmd, 1, I_EndPos
- 1)
 else ; This is a directive/command with no
parameters.
 I_CurrCmd := A_LoopReadLine

 StrReplace, I_CurrCmd, %I_CurrCmd%, [

```

```
StrReplace, I_CurrCmd, %I_CurrCmd%, %A_Space%
StrReplace, I_FullCmd, %I_FullCmd%, ``n, `n
StrReplace, I_FullCmd, %I_FullCmd%, ``t, `t
```

```
; Make arrays of command names and full cmd
syntaxes:
```

```
I_Cmd%A_Index% := I_CurrCmd
I_FullCmd%A_Index% := I_FullCmd
```

```
}
```

```
; Use the Input command to watch for commands that
the user types:
```

```
Loop
```

```
{
```

```
; Editor window check:
```

```
WinGetTitle, ActiveTitle, A
if !InStr(ActiveTitle, I_Editor)
```

```
{
```

```
 ToolTip
 Sleep, 500
 Continue
```

```
}
```

```
; Get all keys till endkey:
```

```
Input, I_Word, V, {enter}{escape}{space}`,
I_EndKey := ErrorLevel
```

```
; Tooltip is hidden in these cases:
```

```
if I_EndKey = "EndKey:Enter" or I_EndKey =
"EndKey:Escape"
```

```
{
```

```
 ToolTip
 Continue
```

```
}
```

```
; Editor window check again!
```

```
WinGetTitle, ActiveTitle, A
```

```
if !InStr(ActiveTitle, I_Editor)
{
 Tooltip
 Continue
}
```

**; Compensate for any indentation that is present:**

```
StrReplace, I_Word, %I_Word%, %A_Space%
StrReplace, I_Word, %I_Word%, %A_Tab%
if I_Word = ""
 Continue
```

**; Check for commented line:**

```
I_Check := SubStr(I_Word, 1, 1)
```

```
if (I_Check = ";" or I_Word = "If") ; "If"
```

**seems a little too annoying to show tooltip for.**

```
Continue
```

**; Match word with command:**

```
I_Index := ""
```

```
Loop
```

```
{
```

**; It helps performance to resolve dynamic variables only once.**

**; In addition, the value put into I\_ThisCmd is also used by the**

**; I\_HelpHotkey subroutine:**

```
I_ThisCmd := I_Cmd%A_Index%
```

```
if I_ThisCmd = ""
```

```
 break
```

```
if (I_Word = I_ThisCmd)
```

```
{
```

```
 I_Index := A_Index
```

```
 I_HelpOn := I_ThisCmd
```

```
 break
```

```
}
```

```

}

; If no match then resume watching user input:
if I_Index = ""
 Continue

; Show matched command to guide the user:
I_ThisFullCmd := I_FullCmd%I_Index%
ToolTip, %I_ThisFullCmd%, %A_CaretX%,
%A_CaretY% + 20
}

I_HelpHotkey:
WinGetTitle, ActiveTitle, A
if !InStr(ActiveTitle, I_Editor), Return

ToolTip ; Turn off syntax helper since there is
no need for it now.

SetTitleMatchMode, 1 ; In case it's 3. This
setting is in effect only for this thread.
if !WinExist("AutoHotkey Help")
{
 if !FileExist(ahk_help_file)
 {
 MsgBox, Could not find the help file:
%ahk_help_file%.
 return
 }
 Run, %ahk_help_file%
 WinWait, AutoHotkey Help
}

if I_ThisCmd = "" ; Instead, use what was most
recently typed.

```

```
I_ThisCmd := I_Word
```

```
; The above has set the "last found" window which
we use below:
```

```
WinActivate
```

```
WinWaitActive
```

```
StrReplace, I_ThisCmd, %I_ThisCmd%, #, {#} ;
```

```
Replace leading #, if any.
```

```
Send, !n{home}+{end}%I_HelpOn%{enter}
```

```
return
```

# On-Screen Keyboard

Based on the v1 script

by Jon

This script creates a mock keyboard at the bottom of your screen that shows the keys you are pressing in real time. I made it to help me to learn to touch-type (to get used to not looking at the keyboard). The size of the on-screen keyboard can be customized at the top of the script. Also, you can double-click the tray icon to show or hide the keyboard.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
;---- Configuration Section: Customize the size of
the on-screen keyboard and
; other options here.
```

```
; Changing this font size will make the entire on-
screen keyboard get
; larger or smaller:
```

```
k_FontSize := 10
```

```
k_FontName := "Verdana" ; This can be blank to
use the system's default font.
```

```
k_FontStyle := "Bold" ; Example of an
alternative: Italic Underline
```

```
; Names for the tray menu items:
```

```
k_MenuItemHide := "Hide on-screen &keyboard;"
```

```
k_MenuItemShow := "Show on-screen &keyboard;"
```

```
; To have the keyboard appear on a monitor other
than the primary, specify
```

```
; a number such as 2 for the following variable.
Leave it blank to use
```

```
; the primary:
```

```
k_Monitor := ""
```

```
;---- End of configuration section. Don't change
anything below this point
```

```
; unless you want to alter the basic nature of the
script.
```

```
;---- Create a GUI window for the on-screen
keyboard:
```

```
Gui := GuiCreate("-Caption +ToolWindow
+AlwaysOnTop +Disabled")
```

```
Gui.SetFont("s%k_FontSize% %k_FontStyle%",
k_FontName)
```

```
Gui.MarginY := 0, Gui.MarginX := 0
```

```
;---- Alter the tray icon menu:
```

```
fn := Func("k_ShowHide").bind(Gui, k_MenuItemHide,
k_MenuItemShow)
```

```
Menu("Tray", "Add", k_MenuItemHide, fn)
```

```
Menu("Tray", "Add", "&Exit", "k_MenuExit")
```

```
Menu("Tray", "Default", k_MenuItemHide)
```

```
Menu("Tray", "NoStandard")
```

```
;---- Add a button for each key:
```

```
; The keyboard layout you see:
```

```
k_cL := [["`", "1", "2", "3", "4", "5", "6",
"7", "8", "9", "0", "-", "=", "Back "],
["Tab", "Q", "W", "E", "R", "T", "Y",
"U", "I", "O", "P", "[", "]", "\ "],
["Caps", "A", "S", "D", "F", "G", "H",
"J", "K", "L", ";", "'", "Enter"],
["Shift", "Z", "X", "C", "V", "B", "N",
"M", ",", ".", "/", "Shift "],
["Ctrl", "Win", "Alt", "Space ", "Alt",
"Win", "Menu", "Ctrl"]]
```

```

; AutoHotkey's official key names for the keys
above (leave empty if identical):
k_oL := [[, , , , , , , , , , , , , , , "Backspace"]
, []
, ["CapsLock"]
, ["LShift", , , , , , , , , , , , , "RShift"]
, ["LCtrl", "LWin", "LAlt", , , "RAlt",
"RWin", "AppsKey", "RCtrl"]]

; Traverse each key in the list of custom key
names:
For n, k_Row in k_cL
 For i, k_CustomKeyName in k_Row
 {
 k_OfficialKeyName := k_oL[n][i]
 ; Calculate object dimensions based on chosen
font size:
 opt := "h" k_FontSize * 3 " w" k_FontSize *
(StrLen(k_CustomKeyName) + 2) " x+m"
 if i = 1
 opt .= " y+m xm"
 ; When a key is pressed by the user, click the
corresponding button on-screen:
 fn :=
Func("k_KeyPress").bind(Gui.Add("Button", opt,
k_CustomKeyName))
 ; If the key has an official key name use it:
 if k_OfficialKeyName
 Hotkey("~*" k_OfficialKeyName, fn)
 else
 Hotkey("~*" Trim(k_CustomKeyName), fn)
 }

;---- Position the keyboard at the bottom of the
screen (taking into account
; the position of the taskbar):

```

```

Gui.Show("Hide") ; Required to get the window's
calculated width and height.
; Calculate window's X-position:
MonitorGetWorkArea(k_Monitor, WL,, WR, WB)
k_xPos := (WR - WL - Gui.Pos.W) / 2 ; Calculate
position to center it horizontally.
; The following is done in case the window will be
on a non-primary monitor
; or if the taskbar is anchored on the left side
of the screen:
k_xPos += WL
; Calculate window's Y-position:
k_yPos := WB - Gui.Pos.H

;---- Show the window:
Gui.Show("x" k_xPos " y" k_yPos " NA")

;---- Function definitions:
k_KeyPress(BtnCtrl)
{
 BtnCtrl.Opt("Default") ; Highlight the last
pressed key.
 ControlClick(, "ahk_id %BtnCtrl.Hwnd%",,,, "D")
 KeyWait(SubStr(A_ThisHotkey, 3))
 ControlClick(, "ahk_id %BtnCtrl.Hwnd%",,,, "U")
}

k_ShowHide(GuiObj, HideText, ShowText)
{
 static isVisible := true
 if isVisible
 {
 GuiObj.Hide()
 Menu("Tray", "Rename", HideText, ShowText)
 isVisible := false
 }
 else

```



# Minimize Window to Tray Menu

This script assigns a hotkey of your choice to hide any window so that it becomes an entry at the bottom of the script's tray menu. Hidden windows can then be unhidden individually or all at once by selecting the corresponding item on the menu. If the script exits for any reason, all the windows that it hid will be unhidden automatically.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; CHANGES:
; July 17, 2016
; - Revised code for AHK v2 compatibility
;
; July 22, 2005 (changes provided by egilmour):
; - Added new hotkey to unhide the last hidden
window (Win+U)
;
; November 3, 2004 (changes provided by trogdor):
; - Program manager is prevented from being
hidden.
; - If there is no active window, the minimize-to-
tray hotkey will have
; no effect rather than waiting indefinitely.
;
; October 23, 2004:
; - The taskbar is prevented from being hidden.
; - Some possible problems with long window titles
have been fixed.
; - Windows without a title can be hidden without
causing problems.
; - If the script is running under AHK v1.0.22 or
greater, the
```

```
; maximum length of each menu item is increased
from 100 to 260.
```

```
; CONFIGURATION SECTION: Change the below values
as desired.
```

```
; This is the maximum number of windows to allow
to be hidden (having a
; limit helps performance):
```

```
mwt_MaxWindows := 50
```

```
; This is the hotkey used to hide the active
window:
```

```
mwt_Hotkey := "#h" ; Win+H
```

```
; This is the hotkey used to unhide the last
hidden window:
```

```
mwt_UnHotkey := "#u" ; Win+U
```

```
; If you prefer to have the tray menu empty of all
the standard items,
; such as Help and Pause, use False. Otherwise,
use True:
```

```
mwt_StandardMenu := false
```

```
; These next few performance settings help to keep
the action within the
```

```
; #HotkeyModifierTimeout period, and thus avoid
the need to release and
```

```
; press down the hotkey's modifier if you want to
hide more than one
```

```
; window in a row. These settings are not needed
you choose to have the
```

```
; script use the keyboard hook via
#InstallKeybdHook or other means:
```

```
#HotkeyModifierTimeout 100
```

```
SetWinDelay 10
```

```
SetKeyDelay 0

#SingleInstance ; Allow only one instance of this
script to be running.

; END OF CONFIGURATION SECTION (do not make
changes below this point
; unless you want to change the basic
functionality of the script).

Hotkey, %mwt_Hotkey%, mwt_Minimize
Hotkey, %mwt_UnHotkey%, mwt_UnMinimize

; If the user terminates the script by any means,
unhide all the
; windows first:
OnExit("mwt_RestoreAllThenExit")

if mwt_StandardMenu = true
 Menu, Tray, Add
else
{
 Menu, Tray, NoStandard
 Menu, Tray, Add, E&xit; and Unhide All,
mwt_RestoreAllThenExit
}
Menu, Tray, Add, &Unhide; All Hidden Windows,
mwt_RestoreAll
Menu, Tray, Add ; Another separator line to make
the above more special.

mwt_MaxLength := 260 ; Reduce this to restrict
the width of the menu.

return ; End of auto-execute section.
```

```

mwt_Minimize:
if mwt_WindowCount >= mwt_MaxWindows
{
 MsgBox No more than %mwt_MaxWindows% may be
hidden simultaneously.
 return
}

; Set the "last found window" to simplify and help
performance.
; Since in certain cases it is possible for there
to be no active window,
; a timeout has been added:
WinWait, A,, 2
if ErrorLevel <> 0 ; It timed out, so do nothing.
 return

; Otherwise, the "last found window" has been set
and can now be used:
WinGetID, mwt_ActiveID
WinGetTitle, mwt_ActiveTitle
WinGetClass, mwt_ActiveClass
if mwt_ActiveClass ~= "Shell_TrayWnd|Progman"
{
 MsgBox The desktop and taskbar cannot be
hidden.
 return
}

; Because hiding the window won't deactivate it,
activate the window
; beneath this one (if any). I tried other ways,
but they wound up
; activating the task bar. This way sends the
active window (which is
; about to be hidden) to the back of the stack,
which seems best:
Send, !{esc}

```

```
; Hide it only now that WinGetTitle/WinGetClass
above have been run (since
; by default, those commands cannot detect hidden
windows):
```

```
WinHide
```

```
; If the title is blank, use the class instead.
This serves two purposes:
```

```
; 1) A more meaningful name is used as the menu
name.
```

```
; 2) Allows the menu item to be created
(otherwise, blank items wouldn't
; be handled correctly by the various routines
below).
```

```
if mwt_ActiveTitle = ""
 mwt_ActiveTitle := "ahk_class
%mwt_ActiveClass"
```

```
; Ensure the title is short enough to fit.
mwt_ActiveTitle also serves to
; uniquely identify this particular menu item.
```

```
mwt_ActiveTitle := SubStr(mwt_ActiveTitle, 1,
mwt_MaxLength)
```

```
; In addition to the tray menu requiring that each
menu item name be
```

```
; unique, it must also be unique so that we can
reliably look it up in
```

```
; the array when the window is later unhidden. So
make it unique if it
```

```
; isn't already:
```

```
Loop, mwt_MaxWindows
```

```
{
```

```
 if mwt_WindowTitle%a_index% = mwt_ActiveTitle
```

```
 {
```

```
 ; Match found, so it's not unique.
```

```
 mwt_ActiveIDShort := Format("{:X}"
```

```
,mwt_ActiveID)
```

```

 mwt_ActiveIDShortLength :=
StrLen(mwt_ActiveIDShort)
 mwt_ActiveTitleLength :=
StrLen(mwt_ActiveTitle)
 mwt_ActiveTitleLength +=
mwt_ActiveIDShortLength
 mwt_ActiveTitleLength += 1 ; +1 the 1
space between title & ID.
 if mwt_ActiveTitleLength > mwt_MaxLength
 {
 ; Since menu item names are limited in
length, trim the title
 ; down to allow just enough room for
the Window's Short ID at
 ; the end of its name:
 TrimCount := mwt_ActiveTitleLength
 TrimCount -= mwt_MaxLength
 mwt_ActiveTitle :=
SubStr(mwt_ActiveTitle, 1, -TrimCount)
 }
 ; Build unique title:
 mwt_ActiveTitle := " " mwt_ActiveIDShort
 break
 }
}

; First, ensure that this ID doesn't already exist
in the list, which can
; happen if a particular window was externally
unhidden (or its app unhid
; it) and now it's about to be re-hidden:
mwt_AlreadyExists := false
Loop, mwt_MaxWindows
{
 if mwt_WindowID%a_index% = mwt_ActiveID
 {
 mwt_AlreadyExists := true
 }
}

```

```

 break
 }
}

; Add the item to the array and to the menu:
if mwt_AlreadyExists = false
{
 Menu, Tray, add, %mwt_ActiveTitle%,
RestoreFromTrayMenu
 mwt_WindowCount += 1
 Loop, mwt_MaxWindows ; Search for a free
slot.
 {
 ; It should always find a free slot if
things are designed right.
 if mwt_WindowID%a_index% = "" ; An empty
slot was found.
 {
 mwt_WindowID%a_index% := mwt_ActiveID
 mwt_WindowTitle%a_index% :=
mwt_ActiveTitle
 break
 }
 }
}
return

RestoreFromTrayMenu:
Menu, Tray, delete, %A_ThisMenuItem%
; Find window based on its unique title stored as
the menu item name:
Loop, mwt_MaxWindows
{
 if mwt_WindowTitle%a_index% = A_ThisMenuItem
; Match found.
 {

```

```
IDToRestore := mwt_WindowID%a_index%
WinShow, ahk_id %IDToRestore%
WinActivate ahk_id %IDToRestore% ;
```

Sometimes needed.

```
mwt_WindowID%a_index% := "" ; Make it
blank to free up a slot.
```

```
mwt_WindowTitle%a_index% := ""
mwt_WindowCount -= 1
break
}
}
return
```

; This will pop the last minimized window off the stack and unhide it.

```
mwt_UnMinimize:
```

; Make sure there's something to unhide.

```
if mwt_WindowCount > 0
{
```

; Get the id of the last window minimized and unhide it

```
IDToRestore := mwt_WindowID%mwt_WindowCount%
WinShow, ahk_id %IDToRestore%
WinActivate ahk_id %IDToRestore%
```

; Get the menu name of the last window minimized and remove it

```
MenuToRemove :=
mwt_WindowTitle%mwt_WindowCount%
Menu, Tray, delete, %MenuToRemove%
```

; clean up our 'arrays' and decrement the window count

```
mwt_WindowID%mwt_WindowCount% := ""
mwt_WindowTitle%mwt_WindowCount% := ""
mwt_WindowCount -= 1
```

```

}
return

mwt_RestoreAllThenExit()
{
 Gosub, mwt_RestoreAll
 ExitApp ; Do a true exit.
}

mwt_RestoreAll:
Loop, mwt_MaxWindows
{
 if mwt_WindowID%a_index% <> ""
 {
 IDToRestore := mwt_WindowID%a_index%
 WinShow, ahk_id %IDToRestore%
 WinActivate ahk_id %IDToRestore% ;
 Sometimes needed.
 ; Do it this way vs. DeleteAll so that the
 sep. line and first
 ; item are retained:
 MenuToRemove := mwt_WindowTitle%a_index%
 Menu, Tray, delete, %MenuToRemove%
 mwt_WindowID%a_index% := "" ; Make it
 blank to free up a slot.
 mwt_WindowTitle%a_index% := ""
 mwt_WindowCount -= 1
 }
 if mwt_WindowCount = 0
 break
}
return

```

# Seek Based on the v1 script by Phi

Navigating the Start Menu can be a hassle, especially if you have installed many programs over time. 'Seek' lets you specify a case-insensitive key word/phrase that it will use to filter only the matching programs and directories from the Start Menu, so that you can easily open your target program from a handful of matched entries. This eliminates the drudgery of searching and traversing the Start Menu.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
/*
```

```
Options:
```

```
-cache Use the cached directory-listing if
available (this is the default mode
when no option is specified)
```

```
-scan Force a directory scan to retrieve the
latest directory listing
```

```
-scex Scan & exit (this is useful for scheduling
the potentially
time-consuming directory-scanning as a
background job)
```

```
-help Show this help
```

```
Important notes:
```

```
Check AutoHotkey's Tutorial how to run this
script, or to compile it if
necessary.
```

```
The only 2 files 'Seek' creates are placed in your
```

TMP directory:

- a. `_Seek.key` (cache file for last query string)
- b. `_Seek.list` (cache file for directory listing)

When you run 'Seek' for the first time, it'll scan your Start Menu, and save the directory listing into a cache file.

The following directories are included in the scanning:

- `A_StartMenu`
- `A_StartMenuCommon`

By default, subsequent runs will read from the cache file so as to reduce the loading time. For more info on options, run '`Seek.exe -help`'. If you think your Start Menu doesn't contain too many programs, you can choose not to use the cache and instruct 'Seek' to always do a directory scan (via option `-scan`). That way, you will always get the latest listing.

When you run 'Seek', a window will appear, waiting for you to enter a key word/phrase. After you have entered a query string, a list of matching records will be displayed. Next, you need to highlight an entry and press ENTER or click on the 'Open' button to run the selected program or open the selected directory.

`* /`

`/*`

Specify which program to use when opening a

directory. If the program cannot be found or is not specified (i.e. variable is unassigned or assigned a null value), the default Explorer will be used.

\*/

```
config := {dirExplorer:
"E:\utl\xplorer2_lite\xplorer2.exe"}
```

/\*

User's customised list of additional directories to be included in the scanning. The full path must not be enclosed by quotes or double-quotes.

If this file is missing, only the default directories will be scanned.

\*/

```
config.SeekMyDir := "%A_ScriptDir%\Seek.dir"
```

/\*

Specify the filename and directory location to save the cached directory/program listing and the cached key word/phrase of the last search.

There is no need to change this unless you want to.

\*/

```
config.saveFile := "%A_Temp%_Seek.ini"
```

/\*

Track search string (True/False)

If true, the last-used query string will be re-used as the default query

string the next time you run Seek. If false, the last-used query string will

not be tracked and there will not be a default query string value the

next time you run Seek.

```

*/
config.TrackKeyPhrase := True

/*
Specify what should be included in scan.
F: Files are included.
D: Directories are included.
*/
config.ScanMode := "FD"

; Init:
; #NoTrayIcon

; Define the script title:
config.ScriptTitle := "Seek - Search the Start
Menu"

; Display the help instructions:
if A_Args[1] ~= "^(--help|-help|/h|-h|/\?|-\?)$"
{
 MsgBox("
 (
 Navigating the Start Menu can be a hassle,
 especially if you have installed
 many programs over time. 'Seek' lets you
 specify a case-insensitive key
 word/phrase that it will use to filter only
 the matching programs and
 directories from the Start Menu, so that you
 can easily open your target
 program from a handful of matched entries.
 This eliminates the drudgery of
 searching and traversing the Start Menu.

 Options:
 -cache Use the cached directory-listing if
 available (this is the default mode

```

```
 when no option is specified)
 -scan Force a directory scan to retrieve
the latest directory listing
 -scex Scan & exit (this is useful for
scheduling the potentially
 time-consuming directory-scanning as a
background job)
 -help Show this help
)", config.ScriptTitle)
 ExitApp()
}
```

```
; Check that the mandatory environment variables
exist and are valid:
```

```
if !FileExist(A_Temp) ; Path does not exist.
{
 MsgBox("
 (
 This mandatory environment variable is either
not defined or invalid:

 TMP = %A_Temp%

 Please fix it before running Seek.
)", config.ScriptTitle)
 ExitApp()
}
```

```
; Scan the Start Menu without Gui:
```

```
if A_Args[1] = "-scex"
{
 SaveFileList(config)
 return
}
```

```
; Create the GUI window:
```

```
G := GuiCreate(, config.ScriptTitle)
```

```
; Add the text box for user to enter the query string:
```

```
E_Search := G.Add("Edit", "W600")
if config.TrackKeyPhrase
 E_Search.Value := IniRead(config.saveFile,
 "LastSession", "SearchText")
```

```
; Add my fav tagline:
```

```
G.Add("Text", "X625 Y10", "What do you seek, my
friend?")
```

```
; Add the status bar for providing feedback to user:
```

```
T_Info := G.Add("Text", "X10 Y31 R1 W764")
```

```
; Add the selection listbox for displaying search results:
```

```
LB := G.Add("ListBox", "X10 Y53 R28 W764 HScroll
Disabled")
```

```
; Add these buttons, but disable them for now:
```

```
B_1 := G.Add("Button", "Default X10 Y446
Disabled", "Open")
B_2 := G.Add("Button", "X59 Y446 Disabled", "Open
Directory")
B_3 := G.Add("Button", "X340 Y446", "Scan Start-
Menu")
```

```
; Add the Exit button:
```

```
B_4 := G.Add("Button", "X743 Y446", "Exit")
```

```
; Create function references for event callbacks:
```

```
FindMatches := Func("FindMatches").bind(config,
LB, B_1, B_2)
OpenTarget := Func("OpenTarget").bind(config, LB)
OpenFolder := Func("OpenFolder").bind(config, LB)
```

```
ScanStartMenu :=
Func("ScanStartMenu").bind(config, E_Search,
T_Info, LB, B_1, B_2, B_3)
Gui_Close := Func("Gui_Close").bind(config,
E_Search, LB)
```

#### **; Add control events:**

```
E_Search.OnEvent("Change", FindMatches)
LB.OnEvent("DoubleClick", OpenTarget)
B_1.OnEvent("Click", OpenTarget)
B_2.OnEvent("Click", OpenFolder)
B_3.OnEvent("Click", ScanStartMenu)
B_4.OnEvent("Click", Gui_Close)
```

#### **; Add window events:**

```
G.OnEvent("Close", Gui_Close)
G.OnEvent("Escape", Gui_Close)
```

#### **; Pop-up the query window:**

```
G.Show("Center")
```

#### **; Force re-scanning if -scan is enabled or listing cache file does not exist:**

```
if (A_Args[1] = "-scan" or
!FileExist(config.saveFile))
 ScanStartMenu(config, E_Search, T_Info, LB, B_1,
B_2, B_3)
```

#### **; Retrieve an set the matching list:**

```
FindMatches(config, LB, B_1, B_2, E_Search)
```

#### **; Retrieve the last selection from cache file and select the item:**

```
if config.TrackKeyPhrase
 LB.Choose(IniRead(config.saveFile,
"LastSession", "Selection"))
```

```

; Function definitions ---

; Scan the start-menu and store the
directory/program listings in a cache file:
ScanStartMenu(config, E_Search, T_Info, LB, B_1,
B_2, B_3)
{
; Inform user that scanning has started:
T_Info.Value := "Scanning directory listing..."

; Disable listbox while scanning is in progress:
LB.Enabled := false
B_1.Enabled := false
B_2.Enabled := false
B_3.Enabled := false

; Retrieve and save the start menu files:
SaveFileList(config)

; Inform user that scanning has completed:
T_Info.Value := "Scan completed."

; Enable listbox:
LB.Enabled := true
B_1.Enabled := true
B_2.Enabled := true
B_3.Enabled := true

; Filter for search string with the new listing:
FindMatches(config, LB, B_1, B_2, E_Search)
}

; Retrieve and save the start menu files:
SaveFileList(config)
{
; Define the directory paths to retrieve:
LocationArray := [A_StartMenu,

```

```
A_StartMenuCommon]
```

```
; Include additional user-defined paths for scanning:
```

```
if FileExist(config.SeekMyDir)
 LoopRead(config.SeekMyDir)
 {
 if !FileExist(A_LoopReadLine)
 MsgBox("
 (
 Processing your customised directory
list...

 '%A_LoopReadLine%' does not exist and
will be excluded from the scanning.
 Please update [%config.SeekMyDir%].
)", config.ScriptTitle, 8192)
 else
 LocationArray.Push(A_LoopReadLine)
 }
```

```
; Scan directory listing by recursing each directory to retrieve the contents.
```

```
; Hidden files are excluded:
```

```
IniDelete(config.saveFile, "LocationList")
For i, Location in LocationArray
{
; Save space by using relative paths:
 IniWrite(Location, config.saveFile,
"LocationList", "L" i)
 A_WorkingDir := Location
 LoopFiles("*", config.ScanMode "R")
 if !InStr(FileGetAttrib(A_LoopFilePath),
"H") ; Exclude hidden file.
 FileList .= "`%L" i "%\" A_LoopFilePath
 "`n"
}
```

```
IniDelete(config.saveFile, "FileList")
IniWrite(FileList, config.saveFile, "FileList")
}
```

```
; Search and display all matching records in the
listbox:
```

```
FindMatches(config, LB, B_1, B_2, E_Search)
```

```
{
```

```
FileArray := []
```

```
SearchText := E_Search.Value
```

```
; Filter matching records based on user query
string:
```

```
if SearchText
```

```
{
```

```
Loop
```

```
{
```

```
Location := IniRead(config.saveFile,
"LocationList", "L" A_Index)
```

```
if !Location
```

```
break
```

```
L%A_Index% := Location
```

```
}
```

```
LoopParse(IniRead(config.saveFile,
"FileList"), "`n")
```

```
{
```

```
Line := Deref(A_LoopField) ; Replace %L_n%
with location paths.
```

```
if SearchText <gt; E_Search.Value
```

```
{
```

```
; User has changed the search string.
```

```
; There is no point to continue searching
using the old string, so abort.
```

```
return
```

```
}
```

```
else
```

```
{
```

```
; Append matching records into the list:
```

```

SplitPath(Line, Name)
MatchFound := true
LoopParse(SearchText, " ")
{
 if !InStr(Name, A_LoopField)
 {
 MatchFound := false
 break
 }
}
if MatchFound
 FileArray.Push(Line)
}
}
}

```

```

; Refresh list with search results:

```

```

LB.Delete(), LB.Add(FileArray)

```

```

if !FileArray.Length()
{

```

```

; No matching record is found. Disable
listbox:

```

```

 LB.Enabled := false
 B_1.Enabled := false
 B_2.Enabled := false

```

```

}
else

```

```

; Matching records are found. Enable listbox:

```

```

 LB.Enabled := true
 B_1.Enabled := true
 B_2.Enabled := true

```

```

; Select the first record if no other record
has been selected:

```

```

 if LB.Text = ""
 LB.Choose(1, 1)

```

```

}
}

; User clicked on 'Open' button or pressed ENTER:
OpenTarget(config, LB)
{
 ; Selected record does not exist (file or
 directory not found):
 if !FileExist(LB.Text)
 {
 MsgBox("
 (
 %LB.Text% does not exist.

 This means that the directory cache is
 outdated. You may click on
 the 'Scan Start-Menu' button below to update
 the directory cache with your
 latest directory listing now.
)", config.ScriptTitle, 8192)
 return
 }

 ; Check whether the selected record is a file or
 directory:
 fileAttrib := FileGetAttrib(LB.Text)
 if InStr(fileAttrib, "D") ; is directory
 OpenFolder(config, LB)
 else if fileAttrib ; is file
 Run(LB.Text)
 else
 {
 MsgBox("
 (
 %LB.Text% is neither a DIRECTORY or a FILE.

 This shouldn't happen. Seek cannot proceed.

```

```

Quitting...
)")
}
WinClose()
}

; User clicked on 'Open Directory' button:
OpenFolder(config, LB)
{
 Path := LB.Text
 ; If user selected a file-record instead of a
 directory-record, extract the
 ; directory path (I'm using DriveGet instead of
 FileGetAttrib to allow the
 ; scenario whereby LB.Text is invalid but the
 directory path of LB.Text is valid):
 if DriveGet("Status", Path) <gt; "Ready" ; not a
 directory
 {
 SplitPath(Path,, Dir)
 Path := Dir
 }

 ; Check whether directory exists:
 if !FileExist(Path)
 {
 MsgBox("
 (
 %Path% does not exist.

 This means that the directory cache is
 outdated. You may click on
 the 'Scan Start-Menu' button below to update
 the directory cache with your
 latest directory listing now.
)", config.ScriptTitle, 8192)
 return
 }
}

```

```

}

; Open the directory:
if FileExist(config.dirExplorer)
 Run('"%config.dirExplorer%" "%Path%") ; Open
with custom file explorer.
else
 Run(Path) ; Open with default windows file
explorer.
}

Gui_Close(config, E_Search, LB)
{
; Save the key word/phrase for next run:
if config.TrackKeyPhrase
{
 IniWrite(E_Search.Value, config.saveFile,
"LastSession", "SearchText")
 IniWrite(LB.Text, config.saveFile,
"LastSession", "Selection")
}
ExitApp()
}

```

# ToolTip Mouse Menu (requires XP/2k/NT) -- by Rajat

This script displays a popup menu in response to briefly holding down the middle mouse button. Select a menu item by left-clicking it. Cancel the menu by left-clicking outside of it. A recent improvement is that the contents of the menu can change depending on which type of window is active (Notepad and Word are used as examples here).

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; You can set any title here for the menu:
MenuTitle := "-----"

; This is how long the mouse button must be held
to cause the menu to appear:
UMDelay := 20

#SingleInstance

; _____
; _____Menu Definitions_____

; Create / Edit Menu Items here.
; You can't use spaces in keys/values/section
names.

; Don't worry about the order, the menu will be
sorted.

MenuItems := "Notepad/Calculator/Section 3/Section
```

4/Section 5"

```
; _____
; _____
; _____Dynamic menuitems here_____
```

```
; Syntax:
; Dyn# = MenuItem|Window title
```

```
Dyn1 := "MS Word|- Microsoft Word"
Dyn2 := "Notepad II|- Notepad"
```

```
; _____
```

Exit

```
; _____
; _____
; _____Menu Sections_____
```

```
; Create / Edit Menu Sections here.
```

```
Notepad:
Run, Notepad.exe
Return
```

```
Calculator:
Run, Calc
Return
```

```
Section3:
MsgBox, You selected 3
Return
```

```
Section4:
MsgBox, You selected 4
Return
```

```
Section5:
MsgBox, You selected 5
Return
```

```
MSWord:
msgbox, this is a dynamic entry (word)
Return
```

```
NotepadII:
msgbox, this is a dynamic entry (notepad)
Return
```

```
;

; _____Hotkey Section_____
```

```
~MButton::
HowLong := 0
Loop
{
 HowLong++
 Sleep, 10
 if !GetKeyState("MButton", "P")
 Break
}
if HowLong < UMDelay, return
```

```
;

;prepares dynamic menu
```

```
DynMenu := ""
Loop
{
 if Dyn%a_index% = "", break

 ppos := InStr(dyn%a_index%, "|")
 item := SubStr(dyn%a_index%, 1, ppos)
```

```
 ppos += 2
 win := SubStr(dyn%a_index%, ppos, 1000)

 if WinActive(win)
 DynMenu .= "/" item
 }
```

```
;Joins sorted main menu and dynamic menu
TempMenu := Sort(MenuItems, "D/") DynMenu
```

```
;clears earlier entries
Loop
{
 if MenuItem%a_index% = "", break
 MenuItem%a_index% := ""
}
```

```
;creates new entries
Loop, Parse, %TempMenu%, /
{
 MenuItem%a_index% := a_loopfield
}
```

```
;creates the menu
Menu := MenuTitle
Loop
{
 if MenuItem%a_index% = "", break
 numItems++
 MenuText := MenuItem%a_index%
 Menu .= "`n" MenuText
}
```

```
MouseGetPos, mX, mY
HotKey, ~LButton, MenuClick
```

```
HotKey, ~LButton, On
ToolTip, %Menu%, %mX%, %mY%
WinActivate, %MenuTitle%
Return

MenuClick:
HotKey, ~LButton, Off
if !WinActive(MenuTitle)
{
 ToolTip
 Return
}

MouseGetPos, mX, mY
ToolTip
mY -= 3 ;space after which first line
starts
mY /= 13 ;space taken by each line
if mY < 1, return
if mY > numItems, return
TargetSection := MenuItem%Round(mY)%
StrReplace, TargetSection, %TargetSection%,
%a_space%
Gosub, %TargetSection%
Return
```

# Volume On-Screen-Display

Based on the v1

script by Rajat

This script assigns hotkeys of your choice to raise and lower the master and/or wave volume. Both volumes are displayed as different color bar graphs.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; --- User Settings ---
```

```
; The percentage by which to raise or lower the
volume each time:
```

```
config := {Step: 4}
```

```
; How long to display the volume level bar graphs:
```

```
config.DisplayTime := 2000
```

```
; Master Volume Bar color (see the help file to
use more precise shades):
```

```
config.CBM := "Red"
```

```
; Wave Volume Bar color:
```

```
config.CBW := "Blue"
```

```
; Background color:
```

```
config.CW := "Silver"
```

```
; Bar's screen position. Use "center" to center
the bar in that dimension:
```

```
config.PosX := "center"
```

```
config.PosY := "center"
```

```
config.Width := 150 ; width of bar
```

```
config.Thick := 12 ; thickness of bar
```

```

; If your keyboard has multimedia buttons for
Volume, you can
; try changing the below hotkeys to use them by
specifying
; Volume_Up, ^Volume_Up, Volume_Down, and
^Volume_Down:
config.MasterUp := "#Up" ; Win+UpArrow
config.MasterDown := "#Down"
config.WaveUp := "+#Up" ; Shift+Win+UpArrow
config.WaveDown := "+#Down"

; --- Auto Execute Section ---

; DON'T CHANGE ANYTHING HERE (unless you know what
you're doing).

#SingleInstance

; Create the Progress window:
G := GuiCreate("+ToolWindow -Caption -Border
+Disabled")
G.MarginX := 0, G.MarginY := 0
opt := "w" config.Width " h" config.Thick "
background" config.CW
G.Add("Progress", opt " vMaster c" config.CBM)
G.Add("Progress", opt " vWave c" config.CBW)

; Create function references for the hotkeys:
vol_MasterUp := Func("ChangeVolume").bind(config,
G, "+")
vol_MasterDown :=
Func("ChangeVolume").bind(config, G, "-")
vol_WaveUp := Func("ChangeVolume").bind(config, G,
"+", "Wave")
vol_WaveDown := Func("ChangeVolume").bind(config,
G, "-", "Wave")

```

### **; Register hotkeys:**

```
Hotkey(config.MasterUp, vol_MasterUp)
Hotkey(config.MasterDown, vol_MasterDown)
Hotkey(config.WaveUp, vol_WaveUp)
Hotkey(config.WaveDown, vol_WaveDown)
```

### **; --- Function Definitions ---**

```
ChangeVolume(config, G, Prefix, ComponentType :=
"Master")
{
 SoundSet(Prefix config.Step, ComponentType)
 G.Control["Master"].Value :=
Round(SoundGet("Master"))
 G.Control["Wave"].Value :=
Round(SoundGet("Wave"))
 G.Show("x" config.PosX " y" config.PosY)
 HideWindow := Func("HideWindow").bind(G)
 SetTimer(HideWindow, -config.DisplayTime)
}

HideWindow(G)
{
 G.Hide()
}
```

# Window Shading (roll up a window to its title bar) -- by Rajat

This script reduces a window to its title bar and then back to its original size by pressing a single hotkey. Any number of windows can be reduced in this fashion (the script remembers each). If the script exits for any reason, all "rolled up" windows will be automatically restored to their original heights.

[Download This Script](#) | [Other Sample Scripts](#) | [Home](#)

```
; Set the height of a rolled up window here. The
operating system
; probably won't allow the title bar to be hidden
regardless of
; how low this number is:
ws_MinHeight := 25

; This line will unroll any rolled up windows if
the script exits
; for any reason:
OnExit("ExitSub")
return ; End of auto-execute section

#z:: ; Change this line to pick a different
hotkey.
; Below this point, no changes should be made
unless you want to
; alter the script's basic functionality.
; Uncomment this next line if this subroutine is
to be converted
; into a custom menu item rather than a hotkey.
The delay allows
```

```

; the active window that was deactivated by the
displayed menu to
; become active again:
;Sleep, 200
WinGetID, ws_ID, A
Loop, Parse, %ws_IDList%, |
{
 if A_LoopField = ws_ID
 {
 ; Match found, so this window should be
restored (unrolled):
 ws_Height := ws_Window%ws_ID%
 WinMove, ahk_id %ws_ID%,,,,, %ws_Height%
 StrReplace, ws_IDList, %ws_IDList%,
|ws_ID%
 return
 }
}
WinGetPos,,,, ws_Height, A
ws_Window%ws_ID% := ws_Height
WinMove, ahk_id %ws_ID%,,,,, %ws_MinHeight%
ws_IDList .= "|" ws_ID
return

ExitSub()
{
 Loop, Parse, %ws_IDList%, |
 {
 if A_LoopField = "" ; First field in list
is normally blank.
 continue ; So skip it.
 ws_Height := ws_Window%A_LoopField%
 WinMove, ahk_id %A_LoopField%,,,,,
%ws_Height%
 }
 ExitApp ; Must do this for the OnExit
subroutine to actually Exit the script.

```



# DynaRun

Run code dynamically in new AutoHotkey process.

```
OutputVar := DynaRun(Script [, ScriptName,
ScriptParameters, AlternateExecutable])
```

```
Command Example: DynaRun "Msgbox Test",
"MyDynaRunScript"
```

```
Function Example: PID := DynaRun("MsgBox Test",
"MyDynaRunScript")
```

## Parameters

### OutputVar

The name of the variable in which to store the PID of new process.

### Script

Script to execute in new process as pure text.

### ScriptName

The name of the Script, it will be used for the pipe name and for script name, e.g. you will see it as title in a MsgBox.

### ScriptParameters

Parameters for new script. To use parameters for executable you would need to specify those together with executable in `AlternateExecutable`.

This parameter can be an object which will be recreated in A\_Args using [ObjDump/ObjLoad](#)

### AlternateExecutable

AutoHotkey Executable used to run the script. If you want to pass parameters to executable you would need to use that parameter.

### Related

---

[Run](#), [RunAs](#), [ProcessExist](#)

### Example

---

```
DynaRun("MsgBox `%
A_Args.1", "MyDynaRun", "Test") ; Will display
MsgBox Test in new process.
```

# CryptAES

Encrypt and Decrypt data.

```
OutputVar := CryptAES(AddressOrVar, Size, password [,
EncryptOrDecrypt, Algorithm])
```

```
Function Example: size := CryptAES(var, sz,
"password", true)
```

## Parameters

### OutputVar

The name of the variable in which to store the size of encrypted memory.

### AddressOrVar

Variable or memory address of data to be encrypted or decrypted.

### Size

Size of data to be encrypted or decrypted.

### Password

Password to use for encryption or decryption.

### EncryptOrDecrypt (optional)

True or 1 to encrypt or false or 0 to decrypted. If omitted data is encrypted.

### Algorithm (optional)

Algorithm to use, supported values are 128 for CALG\_AES\_128, 192 for CALG\_AES\_192 or 256 for CALG\_AES\_256. If omitted CALG\_AES\_256 is used.

### Related

[ZipRawMemory](#), [UnZipRawMemory](#)

# EnvUpdate

Notifies the OS and all running applications that [environment variable\(s\)](#) have changed.

## EnvUpdate

|                 |                 |             |
|-----------------|-----------------|-------------|
| <b>Command</b>  | <b>Example:</b> | EnvUpdate   |
| <b>Function</b> | <b>Example:</b> | EnvUpdate() |

## ErrorLevel

This command is able to throw an exception on failure. For more information, see [Runtime Errors](#).

[ErrorLevel](#) is set to 1 if there was a problem or 0 otherwise.

## Remarks

Refreshes the OS environment. Similar effect as logging off and then on again.

For example, after making changes to the following registry key, EnvUpdate could be used to broadcast the change:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session
Manager\Environment
```

## Related

EnvSet, RegWrite

## Example

EnvUpdate

# GetVar

Retrieves a pointer to a variable.

```
OutputVar := GetVar(VarName [, ResolveAlias])
```

```
Function Example: varptr := GetVar(MyVar)
```

## Parameters

### OutputVar

The name of variable in which to store pointer to desired variable.

### VarName

The name of the variable.

### ResolveAlias

1 / true to receive pointer to primary variable, FALSE / NULL / 0 to receive pointer to desired variable even if it is alias for another variable.

## Related

[Struct](#), [ahkgetvar](#), [FindFunc](#), [ahkFindFunc](#), [ahkFindLabel](#)

## Examples

```
global _AHKVar := "{Int64 mContentsInt64,Double
```



# MenuGetHandle

Retrieves the [Win32 menu](#) handle of a menu.

```
Handle := MenuGetHandle(MenuName)
```

## Parameters

### MenuName

The name of an existing menu. Menu names are not case sensitive.

## Remarks

The returned handle is valid only until the Win32 menu is destroyed. Once the menu is destroyed, the operating system may reassign the handle value to any menus subsequently created by the script or any other program. Conditions which can cause the menu to be destroyed are listed under [Win32 Menus](#).

## Related

[Menu](#), [MenuGetName](#)

## Examples

```
Menu MyMenu, Add, Item 1, no
Menu MyMenu, Add, Item 2, no
Menu MyMenu, Add, Item B, no
```

```
; Retrieve the number of items in a menu.
```

```
item_count := DllCall("GetMenuItemCount", "ptr",
MenuGetHandle("MyMenu"))
```

```
; Retrieve the ID of the last item.
```

```
last_id := DllCall("GetMenuItemID", "ptr",
MenuGetHandle("MyMenu"), "int", item_count-1)
```

```
MsgBox("MyMenu has %item_count% items, and its
last item has ID %last_id%.")
```

```
no:
```

```
return
```

# MenuGetName

Retrieves the name of a menu given a handle to its underlying [Win32 menu](#).

```
MenuName := MenuGetName(Handle)
```

## Parameters

### Handle

A Win32 menu handle (HMENU).

## Remarks

Only the menu's current handle is recognized. When the menu's underlying Win32 menu is destroyed and recreated, the handle changes. For details, see [Win32 Menus](#).

## Related

[Menu](#), [MenuGetHandle](#)

# ProcessClose

Closes a process.

```
OutputVar := ProcessClose(PID-or-Name)
```

```
Command Example: ProcessClose "Notepad.exe"
Function Example: PID :=
ProcessClose("Notepad.exe")
```

## OutputVar

The name of the variable in which to store the PID of closed process or 0 / false otherwise (there was no matching process or there was a problem terminating it).

## PID-or-Name

This parameter can be either a number (the PID) or a process name as described below. It can also be left blank to get PID of the script itself.

**PID:** The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the [Run](#) command. Similarly, the PID of a window can be determined with [WinGet](#).

**Name:** The name of a process is usually the same as its executable (without path), e.g. notepad.exe or winword.exe. Since a name might

match multiple running processes, only the first process will be reported.  
The name is not case sensitive.

## Remarks

Since the process will be abruptly terminated -- possibly interrupting its work at a critical point or resulting in the loss of unsaved data in its windows (if it has any) -- this method should be used only if a process cannot be closed by using [WinClose](#) on one of its windows.

## Related

[ProcessExist](#), [ProcessSetPriority](#), [ProcessWait](#), [ProcessWaitClose](#), [Run](#), [WinGet](#), [WinClose](#), [WinKill](#), [WinWait](#), [WinWaitClose](#), [WinExist](#)

## Examples

```
ProcessClose, OurPID, Notepad.exe
```

# ProcessExist

Checks if a process exists and returns its PID.

```
OutputVar := ProcessExist(PID-or-Name)
```

```
Function Example: PID :=
ProcessExist("AutoHotkey.exe")
```

## OutputVar

The name of the variable in which to store the PID of found process.

## PID-or-Name

This parameter can be either a number (the PID) or a process name as described below. It can also be left blank to get PID of the script itself.

**PID:** The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the [Run](#) command. Similarly, the PID of a window can be determined with [WinGet](#).

**Name:** The name of a process is usually the same as its executable (without path), e.g. notepad.exe or winword.exe. Since a name might match multiple running processes, only the first process will be reported. The name is not case sensitive.

## Remarks

Alternative method to retrieve the script's PID is `PID := DllCall("GetCurrentProcessId")`.

## Related

[ProcessClose](#), [ProcessSetPriority](#), [ProcessWait](#), [ProcessWaitClose](#), [Run](#), [WinGet](#), [WinClose](#), [WinKill](#), [WinWait](#), [WinWaitClose](#), [WinExist](#)

## Examples

```
ProcessExist, OurPID ; sets OurPID to the PID of
this running script
```

# ProcessSetPriority

Changes the priority (as seen in Windows Task Manager) of a process.

**ProcessSetPriority** Priority, PID-or-Name

**Command Example:** `ProcessSetPriority "High", "Notepad.exe"`

**Function Example:** `PID := ProcessSetPriority("High", "Notepad.exe")`

## OutputVar

The name of the variable in which to store the PID of process which priority was changed or 0 / false otherwise (there was no matching process or there was a problem setting priority).

## PID-or-Name

This parameter can be either a number (the PID) or a process name as described below. It can also be left blank to get PID of the script itself.

**PID:** The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the [Run](#) command. Similarly, the PID of a window can be determined with [WinGet](#).

**Name:** The name of a process is usually the same as its executable

(without path), e.g. notepad.exe or winword.exe. Since a name might match multiple running processes, only the first process will be reported. The name is not case sensitive.

### Priority

Process priority: L (or Low), B (or BelowNormal), N (or Normal), A (or AboveNormal), H (or High), R (or Realtime).

### Remarks

Note: Any process not designed to run at Realtime priority might reduce system stability if set to that level.

### Related

[ProcessExist](#), [ProcessSetPriority](#), [ProcessWait](#), [ProcessWaitClose](#), [Run](#), [WinGet](#), [WinClose](#), [WinKill](#), [WinWait](#), [WinWaitClose](#), [WinExist](#)

### Examples

```
ProcessSetPriority, High
```

# ProcessWait

Waits for a matching processes to exist.

```
OutputVar := ProcessWait(PID-or-Name [, Seconds])
```

```
Command Example: ProcessWait "AutoHotkey.exe",
3.5
```

```
Function Example: PID :=
ProcessWait("AutoHotkey.exe", 3.5)
```

## OutputVar

The name of the variable in which to store the PID of the found process or 0 if no matching process exists or the command timed out (when Seconds parameter is used).

## PID-or-Name

This parameter can be either a number (the PID) or a process name as described below. It can also be left blank to get PID of the script itself.

**PID:** The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the [Run](#) command. Similarly, the PID of a window can be determined with [WinGet](#).

**Name:** The name of a process is usually the same as its executable

(without path), e.g. notepad.exe or winword.exe. Since a name might match multiple running processes, only the first process will be reported. The name is not case sensitive.

### Seconds (optional)

Seconds to wait (can contain a decimal point). If *Seconds* is omitted, the command will wait indefinitely.

## Related

[ProcessClose](#), [ProcessSetPriority](#), [ProcessExist](#), [ProcessWaitClose](#), [Run](#), [WinGet](#), [WinClose](#), [WinKill](#), [WinWait](#), [WinWaitClose](#), [WinExist](#)

## Examples

```
Processwait, PID, AutoHotkey.exe
```

# ProcessWaitClose

Waits for ALL matching processes to close.

```
OutputVar := ProcessWaitClose(PID-or-Name [, Seconds])
```

```
Command Example: ProcessWaitClose
"AutoHotkey.exe", 3.5
Function Example: PID :=
ProcessWaitClose("AutoHotkey.exe", 3.5)
```

## OutputVar

The name of the variable in which to store the PID of the first matching process that still exists (if Seconds parameter is used) or 0 if all matching processes are closed.

## PID-or-Name

This parameter can be either a number (the PID) or a process name as described below. It can also be left blank to get PID of the script itself.

**PID:** The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the [Run](#) command. Similarly, the PID of a window can be determined with [WinGet](#).

**Name:** The name of a process is usually the same as its executable

(without path), e.g. notepad.exe or winword.exe. Since a name might match multiple running processes, only the first process will be reported. The name is not case sensitive.

### Seconds (optional)

Seconds to wait (can contain a decimal point). If *Seconds* is omitted, the command will wait indefinitely.

## Related

[ProcessClose](#), [ProcessSetPriority](#), [ProcessWait](#), [ProcessExist](#), [Run](#), [WinGet](#), [WinClose](#), [WinKill](#), [WinWait](#), [WinWaitClose](#), [WinExist](#)

## Examples

```
ProcesswaitClose, PID, AutoHotkey.exe
```

# ResGet

Reads a resource from executable file (dll or exe).

```
OutputVar := ResGet(Data, Executable, Name, Type, Language)
```

```
Function Example: Size := ResGet(Data, A_AhkPath, "MYRESOURCE")
```

## Parameters

### OutputVar

The name of the variable in which to store the size of read data.

### Data (ByRef)

ByRef variable in which to store read data.

### Executable

Path to the executable file (dll or exe).

### Name

The name of resource.

### Type (optional)

Type of resource. Default is RCDATA (10).

See [MSDN](#) for default resource types.

## Language (optional)

The language of resource.

## Remarks

If resource data is compressed or encrypted you can use [UnZipRawMemory](#) to decrypt and unzip it.

## Related

[ResPut](#), [ResPutFile](#) [ResExist](#), [ResDelete](#), [ResDllCreate](#), [UnZipRawMemory](#)

## Example

```
ResGet(data,A_AhkPath,"RESGET.AHK","LIB")
UnZipRawMemory(&data;,var)
MsgBox % StrGet(&var;,"UTF-8")
```

# WinGetControls

Retrieves the control names for all controls in a window. If no matching window exists or there are no controls in the window, OutputVar is made blank. Otherwise, each control name consists of its class name followed immediately by its sequence number (ClassNN), as shown by Window Spy.

```
OutputVar := WinGetControls(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Controls := WinGetControls("ahk_class Notepad")
```

**OutputVar**

```
WinGetControls, Controls, ahk_class Notepad
```

## Parameters

### OutputVar

The name of the variable in which to store an [array](#) containing control names.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To

use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Controls are sorted according to their Z-order, which is usually the same order as TAB key navigation if the window supports tabbing. Hidden windows are included only if [DetectHiddenWindows](#) has been turned on.

## Examples

```
for i,ClassNN in WinGetControls("ahk_class
Notepad")
 MsgBox % "ClassNN: " ClassNN
```

# WinGetCount

Retrieves the number of existing windows that match the specified WinTitle, WinText, ExcludeTitle, and ExcludeText (0 if none).

```
OutputVar := WinGetCount([WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

```
Function example: Count :=
WinGetCount("ahk_class Notepad")
```

## Parameters

### OutputVar

The name of the variable in which to store the number of found windows.

### WinTitle (optional)

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: `My File.txt ahk_class Notepad`. The use correct

child window when multiple window with same criteria exist specify  
`ahk_parent %VarContainingID%`. For example: `ahk_class #32770`  
`ahk_parent %MyWinID%`

### WinText (optional)

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if `DetectHiddenText` is ON.

### ExcludeTitle (optional)

Windows whose titles include this value will not be considered.

### ExcludeText (optional)

Windows whose text include this value will not be considered.

## Examples

```
MsgBox % WinGetCount("ahk_class Notepad") "
Notepad windows are visible."
```

# WinGetExStyle

Retrieves an 8-digit hexadecimal number representing exstyle or extended style (respectively) of a window.

```
OutputVar := WinGetExStyle(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Function Example: Style :=
WinGetExStyle("ahk_class Notepad")
```

## Parameters

### OutputVar

The name of the variable in which to store the 8-digit hexadecimal number representing exstyle of a window. *OutputVar* is made blank if there are no matching windows.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id`

`%VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. The use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

See the [styles table](#) for a partial listing of styles. Hidden windows are included only if [DetectHiddenWindows](#) has been turned on.

## Examples

```
WinGet, ExStyle, ExStyle, My Window Title
if (ExStyle & 0x8) ; 0x8 is WS_EX_TOPMOST.
... the window is always-on-top, so perform
appropriate action.
```

# WinGetID

Retrieves the ID of a window.

```
OutputVar := WinGetID(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
WinID := WinGetID("ahk_class Notepad")
```

**OutputVar**

```
WinGetID, WinID, ahk_class Notepad
```

## Parameters

### OutputVar

The name of the variable in which to store the ID of found window. If there are no matching windows, OutputVar is made blank.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group`

GroupName. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Examples

```
MsgBox % WinGetID("ahk_class Notepad")
```

# WinGetIDLast

Similar to [WinGetID](#) but retrieves the ID of the last/bottommost window if there is more than one match. If there is only one match, it performs identically to ID.

```
OutputVar := WinGetIDLast(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
IDLast := WinGetIDLast("ahk_class Notepad")
```

**OutputVar**

```
WinGetIDLast, IDLast, ahk_class Notepad
```

## Parameters

### OutputVar

The name of the variable in which to store the ID of found window. If there are no matching windows, OutputVar is made blank.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid`

`%VarContainingPID%`. To use a [window group](#), specify `ahk_group` `GroupName`. To use a window's [unique ID number](#), specify `ahk_id` `%VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Examples

```
MsgBox % WinGetIDLast("ahk_class Notepad")
```

# WinGetList

Retrieves the unique ID numbers of all existing windows that match the specified WinTitle, WinText, ExcludeTitle, and ExcludeText (to retrieve all windows on the entire system, omit all four title/text parameters).

```
OutputVar := WinGetList([WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

```
Function example: List := WinGetList("ahk_class
Notepad")
```

## Parameters

### OutputVar

The name of the variable in which to store an [array](#) containing window IDs.

### WinTitle (optional)

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id`

`%VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: `My File.txt ahk_class Notepad`. The use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: `ahk_class #32770 ahk_parent %MyWinID%`

### WinText (optional)

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle (optional)

Windows whose titles include this value will not be considered.

### ExcludeText (optional)

Windows whose text include this value will not be considered.

## Remarks

Windows are retrieved in order from topmost to bottommost (according to how they are stacked on the desktop). Hidden windows are included only if [DetectHiddenWindows](#) has been turned on.

## Examples

```
WinGetList, List, ahk_class Notepad
for i, id in WinGetList("ahk_class Notepad")
 MsgBox % "ID: " id
```



# WinGetMinMax

Retrieves the minimized/maximized state for a window.

```
OutputVar := WinGetMinMax(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Funcon example: MinMax :=
WinGetMinMax("ahk_class Notepad")
```

## Parameters

### OutputVar

The name of the variable in which to store the state of the window. *OutputVar* is made blank if no matching window exists; otherwise, it is set to one of the following numbers:

- 1: The window is minimized ([WinRestore](#) can unminimize it).
- 1: The window is maximized ([WinRestore](#) can unmaximize it).
- 0: The window is neither minimized nor maximized.

### WinTitle (optional)

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by [Window Spy](#)). To use a [process identifier \(PID\)](#), specify `ahk_pid`

`%VarContainingPID%`. To use a [window group](#), specify `ahk_group` `GroupName`. To use a window's [unique ID number](#), specify `ahk_id` `%VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText (optional)

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle (optional)

Windows whose titles include this value will not be considered.

### ExcludeText (optional)

Windows whose text include this value will not be considered.

## Examples

```
ActWinState:=WinGetMinMax("A")
If ActWinState>0
 MsgBox Window is Maximized
else if ActWinState<0
 MsgBox Window is Minimized
else MsgBox Window is neither maximized nor
minimized
```

# WinGetProcessName

Retrieves the name of the process (e.g. notepad.exe) that owns a window.

```
OutputVar := WinGetProcessName([WinTitle, WinText,
ExcludeTitle, ExcludeText])
```

```
Name := WinGetProcessName("ahk_class Notepad")
```

**OutputVar**

```
WinGetProcessName, Count, ahk_class Notepad
```

## Parameters

### OutputVar

The name of the variable in which to store the process path of found windows. If there are no matching windows, OutputVar is made blank.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid`

`%VarContainingPID%`. To use a [window group](#), specify `ahk_group` `GroupName`. To use a window's [unique ID number](#), specify `ahk_id` `%VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Examples

```
MsgBox % WinGetProcessName("ahk_class Notepad")
```

# WinGetProcessPath

Retrieves full path of the process (e.g. C:\Windows\notepad.exe) that owns a window.

```
OutputVar := WinGetProcessPath(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
path := WinGetProcessPath("ahk_class Notepad")
```

**OutputVar**

```
WinGetProcessPath, path, ahk_class Notepad
```

## Parameters

### OutputVar

The name of the variable in which to store the process path of found window. If there are no matching windows, OutputVar is made blank.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify ahk\_class ExactClassName (shown by

Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Examples

```
MsgBox % WinGetProcessPath("ahk_class Notepad")
```

# WinGetStyle

Retrieves an 8-digit hexadecimal number representing style or extended style (respectively) of a window.

```
OutputVar := WinGetStyle(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Style := WinGetStyle("ahk_class Notepad")
```

**OutputVar**

```
WinGetStyle, Style ahk_class Notepad
```

## Parameters

### OutputVar

The name of the variable in which to store the 8-digit hexadecimal number representing style of a window. *OutputVar* is made blank if there are no matching windows.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by

Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

See the [styles table](#) for a partial listing of styles. Hidden windows are included only if [DetectHiddenWindows](#) has been turned on.

## Examples

```
WinGetStyle, Style, My Window Title
```

```
if (Style & 0x8000000) ; 0x8000000 is
```

```
WS_DISABLED.
```

```
... the window is disabled, so perform
appropriate action.
```

# WinGetTransColor

Retrieves the color that is marked transparent in a window (see [WinSetTransColor](#) for how to set the TransColor).

```
OutputVar := WinGetTransColor(Colour [, WinTitle,
WinText, ExcludeTitle, ExcludeText])
```

```
Function Example: success :=
WinGetTransColor(0xFFFFFFFF, "ahk_class Notepad")
```

## Parameters

### OutputVar

The name of the variable in which to store true / 1 if Transcolour was set successfully or false / 0 otherwise.

### WinTitle

The title or partial title of the target window (the matching behavior is determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple](#)

*criteria*. For example: *My File.txt ahk\_class Notepad*. The use correct child window when multiple window with same criteria exist specify *ahk\_parent %VarContainingID%*. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Controls are sorted according to their Z-order, which is usually the same order as TAB key navigation if the window supports tabbing. Hidden windows are included only if [DetectHiddenWindows](#) has been turned on.

## Examples

```
MouseGetPos,,, MouseWin
WinGet, TransColor, TransColor, ahk_id %MouseWin%
; TransColor of the window under the mouse cursor.
```

# WinGetTransparent

Retrieves the degree of transparency of a window (see [WinSetTransparent](#) for how to set transparency).

```
OutputVar := WinGetTransparent(WinTitle, WinText,
ExcludeTitle, ExcludeText)
```

```
Function Example: Trans :=
WinGetTransparent("ahk_class Notepad")
```

```
Function Example:
```

## Parameters

### OutputVar

The name of the variable in which to store a number between 0 and 255, where 0 indicates an invisible window and 255 indicates an opaque window. *OutputVar* is made blank if: 1) the OS is older than Windows XP; 2) there are no matching windows; 3) the window has no transparency level; or 4) other conditions (caused by OS behavior) such as the window having been minimized, restored, and/or resized since it was made transparent.

### WinTitle (optional)

The title or partial title of the target window (the matching behavior is

determined by [SetTitleMatchMode](#)). If this and the other 3 window parameters are blank or omitted, the [Last Found Window](#) will be used. To use a window class, specify `ahk_class ExactClassName` (shown by Window Spy). To use a [process identifier \(PID\)](#), specify `ahk_pid %VarContainingPID%`. To use a [window group](#), specify `ahk_group GroupName`. To use a window's [unique ID number](#), specify `ahk_id %VarContainingID%`. The search can be narrowed by specifying [multiple criteria](#). For example: *My File.txt ahk\_class Notepad*. To use correct child window when multiple window with same criteria exist specify `ahk_parent %VarContainingID%`. For example: *ahk\_class #32770 ahk\_parent %MyWinID%*

### WinText (optional)

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle (optional)

Windows whose titles include this value will not be considered.

### ExcludeText (optional)

Windows whose text include this value will not be considered.

## Remarks

Controls are sorted according to their Z-order, which is usually the same order as TAB key navigation if the window supports tabbing. Hidden windows are

included only if [DetectHiddenWindows](#) has been turned on.

## Examples

```
MsgBox % WinGetTransparent("A")

WinGetTransparent,transparency,A
MsgBox % transparency
```

# WinSetAlwaysOnTop

Makes a window stay on top of all other windows.

```
WinSetAlwaysOnTop [OnOffToggle, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetAlwaysOnTop "On", "A"
Function Example: WinSetAlwaysOnTop("On", "A")
```

## Parameters

### OnOffToggle (optional)

[**On|Off|Toggle**]: Use ON to turn on the setting, OFF to turn it off, or TOGGLE to set it to the opposite of its current state. If omitted, it defaults to TOGGLE. The word Topmost can be used in place of AlwaysOnTop.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

`ErrorLevel` is not changed by this command except where indicated above.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

`WinHide`, `WinSetTitle`, `WinMove`, `WinActivate`, `Control`

## Examples

```
WinSetAlwaysOnTop, toggle, Calculator ; Toggle the
always-on-top status of Calculator.
```

# WinSetEnabled

Disables or enables a window (respectively). When a window is disabled, the user cannot move it or interact with its controls. In addition, disabled windows are omitted from the alt-tab list.

```
WinSetEnabled [OnOffToggle, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetEnabled "On", "A"
Function Example: WinSetEnabled("On", "A")
```

## Parameters

### OnOffToggle

Use ON to turn on the setting, OFF to turn it off, or TOGGLE to set it to the opposite of its current state.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

## ExcludeTitle

Windows whose titles include this value will not be considered.

## ExcludeText

Windows whose text include this value will not be considered.

## Remarks

`ErrorLevel` is not changed by this command except where indicated above.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

`WinHide`, `WinSetTitle`, `WinMove`, `WinActivate`, `Control`

## Examples

```
WinSetEnabled, toggle, Calculator ; Toggle the
disabled status
```

# WinSetExStyle

Changes the extended style of a window, respectively.

```
WinSetExStyle Value [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetExStyle "^0x80", "A"
Function Example: WinSetExStyle("^0x80", "A")
```

## Parameters

### Value

If the first character of Value is a plus or minus sign, the extended style(s) in Value are added or removed, respectively. If the first character is a caret (^), the extended style(s) in Value are each toggled to the opposite state. If the first character is a digit, the window's extended style is overwritten completely; that is, it becomes Value.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility).

Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinHide](#), [WinSetTitle](#), [WinMove](#), [WinActivate](#), [Control](#)

## Examples

```
WinSet, ExStyle, ^0x80, WinTitle ; Toggle the
WS_EX_TOOLWINDOW attribute, which removes/adds the
window from the alt-tab list.
```

# WinSetRegion

Changes the shape of a window to be the specified rectangle, ellipse, or polygon.

```
WinSetRegion [Options, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetRegion "AlwaysOnTop",
"On", "A"
```

```
Function Example: WinSetRegion("AlwaysOnTop",
"On", "A")
```

## Parameters

### Options (optional)

If the *Options* parameter is blank, the window is restored to its original/default display area. Otherwise, one or more of the following options can be specified, each separated from the others with space(s):

**Wn:** Width of rectangle or ellipse. For example: `w200`.

**Hn:** Height of rectangle or ellipse. For example: `h300`.

**X-Y:** Each of these is a pair of X/Y coordinates. For example, `200-0` would use 200 for the X coordinate and 0 for the Y.

**E:** Makes the region an ellipse rather than a rectangle. This option is valid only when **W** and **H** are present.

**R[w-h]:** Makes the region a rectangle with rounded corners. For example, `R30-30` would use a 30x30 ellipse for each corner. If **w-h** is omitted,

30-30 is used. **R** is valid only when **W** and **H** are present.

**Rectangle or ellipse:** If the **W** and **H** options are present, the new display area will be a rectangle whose upper left corner is specified by the first (and only) pair of **X-Y** coordinates. However, if the **E** option is also present, the new display area will be an ellipse rather than a rectangle. For example: `WinSet, Region, 50-0 W200 H250 E, WinTitle`.

**Polygon:** When the **W** and **H** options are absent, the new display area will be a polygon determined by multiple pairs of **X-Y** coordinates (each pair of coordinates is a point inside the window relative to its upper left corner). For example, if three pairs of coordinates are specified, the new display area will be a triangle in most cases. The order of the coordinate pairs with respect to each other is sometimes important. In addition, the word **Wind** maybe be present in *Options* to use the winding method instead of the alternating method to determine the polygon's region.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

`ErrorLevel` is set to 1 upon failure and 0 upon success. Failure occurs when: 1) the target window does not exist; 2) one or more *Options* are invalid; 3) more than 2000 pairs of coordinates were specified; or 4) the specified region is invalid or could not be applied to the target window.

Window titles and text are case sensitive. Hidden windows are not detected unless `DetectHiddenWindows` has been turned on.

## Related

[WinHide](#), [WinSetTitle](#), [WinMove](#), [WinActivate](#), [Control](#)

## Examples

```
WinSetRegion, 50-0 W200 H250, WinTitle ; Make all parts of the window outside this rectangle invisible.
```

```
WinSetRegion, 50-0 W200 H250 R40-40, WinTitle ; Same as above but with corners rounded to 40x40.
```

```
WinSetRegion, 50-0 W200 H250 E, WinTitle ; An ellipse instead of a rectangle.
```

```
WinSetRegion, 50-0 250-0 150-250, WinTitle ; A triangle with apex pointing down.
```

```
WinSetRegion,, WinTitle ; Restore the window to
its original/default display area.
```

```
; Here is a region with a more complex area. It
creates a see-through rectangular hole inside a
window.
```

```
; There are two rectangles specified below: an
outer and an inner. Each rectangle consists of 5
pairs
```

```
; of X/Y coordinates because the first pair is
repeated at the end to "close off" each rectangle.
```

```
WinSetRegion, 0-0 300-0 300-300 0-300 0-0 100-
100 200-100 200-200 100-200 100-100, WinTitle
```

# WinSetStyle

Changes the style of a window, respectively.

```
WinSetStyle Value [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetStyle "-0xC00000", "A"
Function Example: WinSetStyle("-0xC00000", "A")
```

## Parameters

### Value

If the first character of Value is a plus or minus sign, the style(s) in Value are added or removed, respectively. If the first character is a caret (^), the style(s) in Value are each toggled to the opposite state. If the first character is a digit, the window's style is overwritten completely; that is, it becomes Value.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility).

Hidden text elements are detected if [DetectHiddenText](#) is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Remarks

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinHide](#), [WinSetTitle](#), [WinMove](#), [WinActivate](#), [Control](#)

## Examples

```
WinSet, Style, -0xC00000, A ; Remove the active
window's title bar (WS_CAPTION).
```

# WinSetTransparentColor

Makes all pixels of the chosen color invisible inside the target window, which allows the contents of the window behind it to show through. If the user clicks on an invisible pixel, the click will "fall through" to the window behind it.

```
WinSetTransparentColor Color [N] [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetTransparentColor "White", "A"
Function Example: WinSetTransparentColor("White",
"A")
```

## Parameters

### Color [N]

Specify a color name or RGB value (see the [color chart](#) for guidance, or use [PixelGetColor](#) in its RGB mode). To additionally make the visible part of the window partially transparent, append a space (not a comma) followed by the transparency level (0-255). For example: `WinSet, TransColor, EEAA99 150, WinTitle`.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if `DetectHiddenText` is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Known Limitations for Transparent and TransColor:

- They have no effect on a window that lacks a caption (title bar) and lacks the `always-on-top` property. For `GUI windows`, this can be solved by removing the caption only after making the window transparent. Alternatively, the following properties allow transparency to take effect: `Gui -Caption +Toolwindow`.
- Setting "Transparent" to 255 prior to turning it off might avoid window redrawing problems such as a black background. If the window still fails to be redrawn correctly, see `Redraw` for a possible workaround.
- To change a window's existing `TransColor`, it may be necessary to turn off transparency before making the change.

**Tip:** To make the task bar transparent, use `WinSet, Transparent, 150, ahk_class Shell_Traywnd`. Similarly, to make the Start Menu transparent, follow this example:

```
DetectHiddenWindows, on
WinSet, Transparent, 150, ahk_class BaseBar ;
```

To make the Start Menu's submenus transparent, also include the script below.

To make all or selected menus on the entire system transparent, keep a script such as the following always running. Note that although such a script cannot make its own menus transparent, it can make those of other scripts transparent:

```
#Persistent
SetTimer, WatchForMenu, 5
return ; End of auto-execute section.

WatchForMenu:
DetectHiddenWindows, on ; Might allow
detection of menu sooner.
IfWinExist, ahk_class #32768
 WinSet, Transparent, 150 ; Uses the window
found by the above line.
return
```

## Remarks

[ErrorLevel](#) is not changed by this command except where indicated above.

Although transparency is supported on Windows 2000/XP or later, retrieving the current transparency settings of a window is possible only on Windows XP or later (via [WinGet](#)).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

[WinGet](#), [WinHide](#), [WinSetTitle](#), [WinMove](#), [WinActivate](#), [Control](#)

## Examples

```
WinSetTransColor, White, Untitled - Notepad ; Make
all white pixels invisible.
```

# WinSetTransparent

Makes a window semi-transparent.

```
WinSetTransparent Value [, WinTitle, WinText,
ExcludeTitle, ExcludeText]
```

```
Command Example: WinSetTransparent 150, "A"
Function Example: WinSetTransparent(150, "A")
```

## Parameters

### Value

Specify a number between 0 and 255 to indicate the degree of transparency: 0 makes the window invisible while 255 makes it opaque. Transparency may be turned off completely for a window by specifying the word OFF. This is different than specifying 255 because it may improve performance and reduce usage of system resources.

### WinTitle

A window title or other criteria identifying the target window. See [WinTitle](#).

### WinText

If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility).

Hidden text elements are detected if `DetectHiddenText` is ON.

### ExcludeTitle

Windows whose titles include this value will not be considered.

### ExcludeText

Windows whose text include this value will not be considered.

## Known Limitations for Transparent and `TransColor`:

- They have no effect on a window that lacks a caption (title bar) and lacks the `always-on-top` property. For GUI windows, this can be solved by removing the caption only after making the window transparent. Alternatively, the following properties allow transparency to take effect: `Gui -Caption +Toolwindow`.
- Setting "Transparent" to 255 prior to turning it off might avoid window redrawing problems such as a black background. If the window still fails to be redrawn correctly, see `Redraw` for a possible workaround.
- To change a window's existing `TransColor`, it may be necessary to turn off transparency before making the change.

**Tip:** To make the task bar transparent, use `WinSet, Transparent, 150, ahk_class Shell_TrayWnd`. Similarly, to make the Start Menu transparent, follow this example:

```
DetectHiddenWindows, on
WinSet, Transparent, 150, ahk_class BaseBar ;
To make the Start Menu's submenus transparent,
```

also include the script below.

To make all or selected menus on the entire system transparent, keep a script such as the following always running. Note that although such a script cannot make its own menus transparent, it can make those of other scripts transparent:

```
#Persistent
SetTimer, WatchForMenu, 5
return ; End of auto-execute section.

WatchForMenu:
DetectHiddenWindows, on ; Might allow
detection of menu sooner.
IfWinExist, ahk_class #32768
 WinSet, Transparent, 150 ; Uses the window
found by the above line.
return
```

## Remarks

[ErrorLevel](#) is not changed by this command except where indicated above.

Although transparency is supported on Windows 2000/XP or later, retrieving the current transparency settings of a window is possible only on Windows XP or later (via [WinGet](#)).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#) has been turned on.

## Related

WinGet, WinHide, WinSetTitle, WinMove, WinActivate, Control

## Examples

```
WinSetTransparent, 200, Untitled - Notepad ; Make
the window a little bit transparent.
```

# The *WinTitle* Parameter & the Last Found Window

Many commands and a few functions have a *WinTitle* parameter, used to identify which window (or windows) to operate on. This parameter can be the title or partial title of the window, and/or any other criteria described on this page.

## Quick Reference:

|              |                    |
|--------------|--------------------|
| <i>Title</i> | Matching Behaviour |
| A            | The Active Window  |
| ahk_class    | Window Class       |
| ahk_id       | Unique ID/HWND     |
| ahk_pid      | Process ID         |
| ahk_exe      | Process Name/Path  |
| ahk_parent   | Parent Window      |
| ahk_group    | Window Group       |
|              | Multiple Criteria  |
| (All empty)  | Last Found Window  |

## Matching Behaviour

`SetTitleMatchMode` controls how a partial or complete title is compared against the title of each window. Depending on the setting, *WinTitle* can be an exact title, or it can contain a prefix, a substring which occurs anywhere in the title, or a `RegEx` pattern. This setting also controls whether `abk_class` is interpreted as an exact class name or a `RegEx` pattern.

Window titles are case sensitive, except when using the `i` modifier in a `RegEx` pattern.

Hidden windows are only detected if `DetectHiddenWindows` is On, except with `WinShow`, which always detects hidden windows.

If multiple windows match the *WinTitle* and any other criteria, the topmost matching window is used. If the active window matches the criteria, it usually takes precedence since it is usually above all other windows. However, if an `always-on-top` window also matches (and the active window is not `always-on-top`), it may be used instead.

## Active Window (A)

If *WinTitle* is the letter **A** and the other 3 window parameters (*WinText*, *ExcludeTitle* and *ExcludeText*) are blank or omitted, the active window will be used.

```
; Retrieve the ID/HWND of the active window
id := WinExist("A")
MsgBox % id

; Press Win+↑ to maximize the active window
#Up::WinMaximize, A
```

## ahk\_class Window Class

A window class is a set of attributes that the system uses as a template to create a window. In other words, the class name of the window identifies what *type* of window it is. To use a window class, specify `ahk_class` `ExactClassName` as shown by Window Spy. *ExactClassName* can be retrieved by `WinGetClass`.

If the RegEx title matching mode is active, `ahk_class` accepts a regular expression.

```
; Activate a console window (e.g. cmd.exe)
WinActivate, ahk_class ConsoleWindowClass
```

## ahk\_id Unique ID / HWND

Each window or control has a unique ID, also known as a HWND (short for handle to window). This ID can be used to identify the window or control even if its title changes. The ID of a window is typically retrieved via [WinExist\(\)](#) or [WinGet](#). The ID of a control is typically retrieved via [ControlGetHwnd](#), [MouseGetPos](#), or [DllCall](#). Also, ahk\_id will operate on controls even if they are hidden; that is, the setting of [DetectHiddenWindows](#) does not matter for controls.

```
WinActivate, ahk_id %VarContainingID%
```

## ahk\_pid Process ID

Use ahk\_pid to identify a window belonging to a specific process. The process identifier (PID) is typically retrieved by [WinGetId](#), [Run](#) or [ProcessExist](#).

```
WinActivate, ahk_pid %VarContainingPID%
```

## ahk\_exe Process Name/Path

Use ahk\_exe to identify a window belonging to any process with the given name or path.

While ahk\_pid is limited to one specific process, ahk\_exe considers all processes with name or full path matching a given string. If the RegEx [title matching mode](#) is active, ahk\_exe accepts a [regular expression](#) which must match the full path of the process. Otherwise, ahk\_exe accepts a case-insensitive name or full path; for example, ahk\_exe notepad.exe covers ahk\_exe C:\Windows\notepad.exe, ahk\_exe C:\Windows\System32\notepad.exe and other variations.

```
; Activate an existing notepad.exe window, or
open a new one
```

```
if WinExist("ahk_exe notepad.exe")
 WinActivate, ahk_exe notepad.exe
else
 Run, notepad.exe
```

```
SetTitleMatchMode RegEx
WinActivate ahk_exe i)\\notepad\.exe$; Match
the name part of the full path.
```

## ahk\_group Window Group

Use `ahk_group` to identify a window or windows matching the rules contained by a previously defined `window group`.

The commands `WinMinimize`, `WinMaximize`, `WinRestore`, `WinHide`, `WinShow`, `WinClose`, and `WinKill` will act on **all** the group's windows. By contrast, the other window commands such as `WinActivate` and `WinExist` will operate only on the topmost window of the group.

```
; Define the group: Windows Explorer windows
GroupAdd, Explorer, ahk_class ExploreWClass ;
Unused on Vista and later
GroupAdd, Explorer, ahk_class CabinetWClass

; Activate any window matching the above
criteria
WinActivate, ahk_group Explorer
```

## Multiple Criteria

By contrast with `ahk_group` (which broadens the search), the search may be narrowed by specifying more than one criterion within the `WinTitle` parameter. In the following example, the script waits for a window whose title contains *My File.txt* **and** whose class is *Notepad*:

```
WinWait My File.txt ahk_class Notepad
WinActivate ; Activate the window it found.
```

When using this method, the text of the title (if any is desired) should be listed first, followed by one or more additional criteria. Criteria beyond the first should be separated from the previous with exactly one space or tab (any other spaces or tabs are treated as a literal part of the previous criterion).

`ahk_id` can be combined with other criteria to test a window's title, class or other attributes:

```
MouseGetPos,,, id
if WinExist("ahk_class Notepad ahk_id " id)
 MsgBox The mouse is over Notepad.
```

## The "Last Found" Window

This is the window most recently found by `IfWin[Not]Exist`, `WinExist`, `IfWin[Not]Active`, `WinActive`, `WinWait[Not]Active`, or `WinWait`. It can make scripts easier to create and maintain since the `WinTitle` and `WinText` of the target window do not need to be repeated for every windowing command. In addition, scripts perform better because they don't need to search for the target window again after it has been found the first time.

The "last found" window can be used by all of the windowing commands except `WinWait`, `WinActivateBottom`, and `GroupAdd`. To use it, simply omit all four window parameters (`WinTitle`, `WinText`, `ExcludeTitle`, and `ExcludeText`).

Each `thread` retains its own value of the "last found" window, meaning that if the `current thread` is interrupted by another, when the original thread is resumed it will still have its original value of the "last found" window, not that of the interrupting thread.

If the last found window is a hidden `Gui window`, it can be used even when `DetectHiddenWindows` is Off. This is often used in conjunction with the GUI's `+LastFound` option.

```
Run Notepad
WinWait Untitled - Notepad
WinActivate ; Uses the last found window.

if WinExist("Untitled - Notepad")
{
```

```
 WinActivate ; Automatically uses the window
found above.
 WinMaximize ; same
 Send, Some text.{Enter}
 return
}

if !WinExist("Calculator")
 return
else
{
 WinActivate ; The above "!WinExist" also set
the "last found" window for us.
 WinMove, 40, 40 ; Move it to a new position.
 return
}
```

# List of Windows Messages

Below is a list of values for the *Msg* parameter of [PostMessage](#) and [SendMessage](#). To discover more about how to use a particular message (e.g. `WM_VSCROLL`), look it up at <http://msdn.microsoft.com> or with a search engine of your choice. Also, check out the [Message Tutorial](#).

```
WM_NULL := 0x00
WM_CREATE := 0x01
WM_DESTROY := 0x02
WM_MOVE := 0x03
WM_SIZE := 0x05
WM_ACTIVATE := 0x06
WM_SETFOCUS := 0x07
WM_KILLFOCUS := 0x08
WM_ENABLE := 0x0A
WM_SETREDRAW := 0x0B
WM_SETTEXT := 0x0C
WM_GETTEXT := 0x0D
WM_GETTEXTLENGTH := 0x0E
WM_PAINT := 0x0F
WM_CLOSE := 0x10
WM_QUERYENDSESSION := 0x11
WM_QUIT := 0x12
WM_QUERYOPEN := 0x13
WM_ERASEBKGND := 0x14
WM_SYSCOLORCHANGE := 0x15
WM_ENDSESSION := 0x16
WM_SYSTEMERROR := 0x17
WM_SHOWWINDOW := 0x18
WM_CTLCOLOR := 0x19
WM_WININICHANGE := 0x1A
WM_SETTINGCHANGE := 0x1A
WM_DEVMODECHANGE := 0x1B
```













# CLSID List (Windows Class Identifiers)

Certain special folders within the operating system are identified by unique strings. Some of these strings can be used with [FileSelect](#), [DirSelect](#), and [Run](#). For example:

```
FileSelect, OutputVar,, ::{645ff040-5081-101b-9f08-00aa002f954e} ; Select a file in the Recycle Bin.
DirSelect, OutputVar, ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ; Select a folder within My Computer.
```

| CLSID                                    | Meaning/Location     | Supported by <a href="#">Run</a> ? |
|------------------------------------------|----------------------|------------------------------------|
| ::{d20ea4e1-3957-11d2-a40b-0c5020524153} | Administrative Tools |                                    |
| ::{85bbd920-42a0-1069-a2e4-08002b30309d} | Briefcase            |                                    |
| ::{21ec2020-3aea-1069-a2dd-08002b30309d} | Control Panel        |                                    |
| ::{d20ea4e1-3957-11d2-a40b-0c5020524152} | Fonts                |                                    |
| ::{ff393560-c2a7-11cf-bff4-444553540000} | History              |                                    |
| ::{00020d75-0000-0000-c000-000000000046} | Inbox                |                                    |
| ::{00028b00-0000-0000-c000-000000000046} | Microsoft Network    |                                    |
| ::{20d04fe0-3aea-1069-a2d8-              | My Computer          | Yes                                |

|                                          |                          |     |
|------------------------------------------|--------------------------|-----|
| 08002b30309d}                            |                          |     |
| ::{450d8fba-ad25-11d0-98a8-0800361b1103} | My Documents             | Yes |
| ::{208d2c60-3aea-1069-a2d7-08002b30309d} | My Network Places        | Yes |
| ::{1f4de370-d627-11d1-ba4f-00a0c91eedba} | Network Computers        | Yes |
| ::{7007acc7-3202-11d1-aad2-00805fc1270e} | Network Connections      | Yes |
| ::{2227a280-3aea-1069-a2de-08002b30309d} | Printers and Faxes       | Yes |
| ::{7be9d83c-a729-4d97-b5a7-1b7313c39e0a} | Programs Folder          |     |
| ::{645ff040-5081-101b-9f08-00aa002f954e} | Recycle Bin              | Yes |
| ::{e211b736-43fd-11d1-9efb-0000f8757fcd} | Scanners and Cameras     |     |
| ::{d6277990-4c6a-11cf-8d87-00aa0060f5bf} | Scheduled Tasks          | Yes |
| ::{48e7caab-b918-4e58-a94d-505519c795dc} | Start Menu Folder        |     |
| ::{7bd29e00-76c1-11cf-9dd0-00a0c9034933} | Temporary Internet Files |     |
| ::{bdeadf00-c265-11d0-bced-00a0c90ab50f} | Web Folders              |     |

The "Yes" entries in the last column are not authoritative: the [Run](#) command might support different CLSIDs depending on system configuration. To open a CLSID folder via Run, simply specify the CLSID as the first parameter. For example:

Run ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ;

Opens the "My Computer" folder.

Run ::{645ff040-5081-101b-9f08-00aa002f954e} ;

Opens the Recycle Bin.

Run ::{450d8fba-ad25-11d0-98a8-0800361b1103}\My

Folder ; Opens a folder that exists inside "My

Documents".

Run %A\_MyDocuments%\My Folder ; Same as the

above on most systems.

# Styles Usable by `GuiCreate`, `Gui object` and `GuiControl object`

## Table of Contents

- [Styles Common to Gui/Parent Windows and Most Control Types](#)
- [Text](#) | [Edit](#) | [UpDown](#) | [Picture](#)
- [Button](#) | [Checkbox](#) | [Radio](#) | [GroupBox](#)
- [DropDownList](#) | [ComboBox](#)
- [ListBox](#) | [ListView](#) | [TreeView](#)
- [DateTime](#) | [MonthCal](#)
- [Slider](#) | [Progress](#) | [Tab](#) | [StatusBar](#)

| Common styles          | Value      | Description                                                                                                      |
|------------------------|------------|------------------------------------------------------------------------------------------------------------------|
| default for GUI window |            | <code>WS_POPUP</code> ,<br><code>WS_CAPTION</code> ,<br><code>WS_SYSMENU</code> ,<br><code>WS_MINIMIZEBOX</code> |
| forced for GUI window  |            | <code>WS_CLIPSIBLINGS</code>                                                                                     |
| <code>WS_BORDER</code> | 0x800000   | +/-Border. Creates a window that has a thin-line border.                                                         |
| <code>WS_POPUP</code>  | 0x80000000 | Creates a pop-up window. This style cannot be used with the <code>WS_CHILD</code> style.                         |
|                        |            | +/-Caption. Creates a                                                                                            |

|                |           |                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WS_CAPTION     | 0xC00000  | window that has a title bar. This style is a numerical combination of <a href="#">WS_BORDER</a> and <a href="#">WS_DLGFRAME</a> .                                                                                                                                                                                                                                                                                                 |
| WS_DISABLED    | 0x8000000 | +/-Disabled. Creates a window that is initially disabled.                                                                                                                                                                                                                                                                                                                                                                         |
| WS_DLGFRAME    | 0x400000  | Creates a window that has a border of a style typically used with dialog boxes.                                                                                                                                                                                                                                                                                                                                                   |
| WS_GROUP       | 0x20000   | +/-Group. Indicates that this control is the first one in a group of controls. This style is automatically applied to manage the "only one at a time" behavior of radio buttons. In the rare case where two groups of radio buttons are added consecutively (with no other control types in between them), this style may be applied manually to the first control of the second radio group, which splits it off from the first. |
| WS_HSCROLL     | 0x100000  | Creates a window that has a horizontal scroll bar.                                                                                                                                                                                                                                                                                                                                                                                |
| WS_MAXIMIZE    | 0x1000000 | Creates a window that is initially maximized.                                                                                                                                                                                                                                                                                                                                                                                     |
| WS_MAXIMIZEBOX | 0x10000   | +/-MaximizeBox. Creates a window that has a maximize button. Cannot be combined with the                                                                                                                                                                                                                                                                                                                                          |

|                     |            |                                                                                                                                                                          |
|---------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |            | WS_EX_CONTEXTHELP style. The WS_SYSMENU style must also be specified.                                                                                                    |
| WS_MINIMIZE         | 0x20000000 | Creates a window that is initially minimized.                                                                                                                            |
| WS_MINIMIZEBOX      | 0x20000    | +/-MinimizeBox. Creates a window that has a minimize button. Cannot be combined with the WS_EX_CONTEXTHELP style. The WS_SYSMENU style must also be specified.           |
| WS_OVERLAPPED       | 0          | Creates an overlapped window. An overlapped window has a title bar and a border. Same as the WS_TILED style.                                                             |
| WS_OVERLAPPEDWINDOW | 0xCF0000   | Creates an overlapped window with the WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX, and WS_MAXIMIZEBOX styles. Same as the WS_TILEDWINDOW style. |
| WS_POPUPWINDOW      | 0x80880000 | Creates a pop-up window with WS_BORDER, WS_POPUP, and WS_SYSMENU styles. The WS_CAPTION and WS_POPUPWINDOW styles must be combined to make the window menu               |

|               |            |                                                                                                                                                                                                                        |
|---------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |            | visible.                                                                                                                                                                                                               |
| WS_SIZEBOX    | 0x40000    | +/--Resize. Creates a window that has a sizing border. Same as the <a href="#">WS_THICKFRAME</a> style.                                                                                                                |
| WS_SYSMENU    | 0x80000    | +/--SysMenu. Creates a window that has a window menu on its title bar. The <a href="#">WS_CAPTION</a> style must also be specified.                                                                                    |
| WS_TABSTOP    | 0x10000    | +/--Tabstop. Specifies a control that can receive the keyboard focus when the user presses the TAB key. Pressing the TAB key changes the keyboard focus to the next control with the <a href="#">WS_TABSTOP</a> style. |
| WS_THICKFRAME | 0x40000    | Creates a window that has a sizing border. Same as the <a href="#">WS_SIZEBOX</a> style                                                                                                                                |
| WS_VSCROLL    | 0x200000   | Creates a window that has a vertical scroll bar.                                                                                                                                                                       |
| WS_VISIBLE    | 0x10000000 | Creates a window that is initially visible.                                                                                                                                                                            |
| WS_CHILD      | 0x40000000 | Creates a child window. A window with this style cannot have a menu bar. This style cannot be used with the <a href="#">WS_POPUP</a> style.                                                                            |



| <a href="#">Text</a> styles | Value | Description |
|-----------------------------|-------|-------------|
|-----------------------------|-------|-------------|

|                |      |                                                                                                                                                                                                                 |
|----------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default        |      | None                                                                                                                                                                                                            |
| forced         |      | None                                                                                                                                                                                                            |
| SS_BLACKFRAME  | 0x7  | Specifies a box with a frame drawn in the same color as the window frames. This color is black in the default color scheme.                                                                                     |
| SS_BLACKRECT   | 0x4  | Specifies a rectangle filled with the current window frame color. This color is black in the default color scheme.                                                                                              |
| SS_CENTER      | 0x1  | +/-Center. Specifies a simple rectangle and centers the error value text in the rectangle. The control automatically wraps words that extend past the end of a line to the beginning of the next centered line. |
| SS_ETCHEDFRAME | 0x12 | Draws the frame of the static control using the EDGE_ETCHED edge style.                                                                                                                                         |
| SS_ETCHEDHORZ  | 0x10 | Draws the top and bottom edges of the static control using the EDGE_ETCHED edge style.                                                                                                                          |
| SS_ETCHEDVERT  | 0x11 | Draws the left and right edges of the static control using the EDGE_ETCHED edge style.                                                                                                                          |
| SS_GRAYFRAME   | 0x8  | Specifies a box with a frame drawn with the same color as the screen background (desktop). This color is gray in the default color scheme.                                                                      |
| SS_GRAYRECT    | 0x5  | Specifies a rectangle filled with the current screen background color. This color is gray in the default color scheme.                                                                                          |

|                   |        |                                                                                                                                                                                                                                                                                                                                              |
|-------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SS_LEFT           | 0      | +/-Left. This is the default. It specifies a simple rectangle and left-aligns the text in the rectangle. The text is formatted before it is displayed. Words that extend past the end of a line are automatically wrapped to the beginning of the next left-aligned line. Words that are longer than the width of the control are truncated. |
| SS_LEFTNOWORDWRAP | 0xC    | +/-Wrap. Specifies a rectangle and left-aligns the text in the rectangle. Tabs are expanded, but words are not wrapped. Text that extends past the end of a line is clipped.                                                                                                                                                                 |
| SS_NOPREFIX       | 0x80   | Prevents interpretation of any ampersand (&) characters in the control's text as accelerator prefix characters. This can be useful when file names or other strings that might contain an ampersand (&) must be displayed within a text control                                                                                              |
| SS_NOTIFY         | 0x100  | Sends the parent window the STN_CLICKED notification when the user clicks the control.                                                                                                                                                                                                                                                       |
| SS_RIGHT          | 0x2    | +/-Right. Specifies a rectangle and right-aligns the specified text in the rectangle.                                                                                                                                                                                                                                                        |
| SS_SUNKEN         | 0x1000 | Draws a half-sunken border around a static control.                                                                                                                                                                                                                                                                                          |
| SS_WHITEFRAME     | 0x9    | Specifies a box with a frame drawn with the same color as the window background. This color is white in the default color scheme.                                                                                                                                                                                                            |

|              |     |                                                                                                                         |
|--------------|-----|-------------------------------------------------------------------------------------------------------------------------|
| SS_WHITERECT | 0x6 | Specifies a rectangle filled with the current window background color. This color is white in the default color scheme. |
|--------------|-----|-------------------------------------------------------------------------------------------------------------------------|



| Edit styles    | Value | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default        |       | <p><a href="#">WS_TABSTOP</a> and <a href="#">WS_EX_CLIENTEDGE</a> (extended style E0x200)</p> <p>If an Edit is auto-detected as multi-line due to its starting contents containing multiple lines, its height being taller than 1 row, or its row-count having been explicitly specified as greater than 1, the following styles will be applied by default: <a href="#">WS_VSCROLL</a>, <a href="#">ES_WANTRETURN</a>, and <a href="#">ES_AUTOVSCROLL</a></p> <p>If an Edit is auto-detected as a single line, it defaults to having <a href="#">ES_AUTOHSCROLL</a>.</p> |
| forced         |       | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ES_AUTOHSCROLL | 0x80  | +/-Wrap for multi-line edits, and +/-Limit for single-line edits. Automatically scrolls text to the right by 10 characters when the user types a character at the end of the line. When the user presses the ENTER key, the control scrolls all text back to the zero position.                                                                                                                                                                                                                                                                                            |
| ES_AUTOVSCROLL | 0x40  | Scrolls text up one page when the user presses the ENTER key on the last line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|               |        |                                                                                                                                                                                                                                                                                                                              |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ES_CENTER     | 0x1    | +/-Center. Centers text in a multiline edit control.                                                                                                                                                                                                                                                                         |
| ES_LOWERCASE  | 0x10   | +/-Lowercase. Converts all characters to lowercase as they are typed into the edit control.                                                                                                                                                                                                                                  |
| ES_NOHIDSEL   | 0x100  | Negates the default behavior for an edit control. The default behavior hides the selection when the control loses the input focus and inverts the selection when the control receives the input focus. If you specify <code>ES_NOHIDSEL</code> , the selected text is inverted, even if the control does not have the focus. |
| ES_NUMBER     | 0x2000 | +/-Number. Prevents the user from typing anything other than digits in the control.                                                                                                                                                                                                                                          |
| ES_OEMCONVERT | 0x400  | This style is most useful for edit controls that contain file names.                                                                                                                                                                                                                                                         |
| ES_MULTILINE  | 0x4    | +/-Multi. Designates a multiline edit control. The default is a single-line edit control.                                                                                                                                                                                                                                    |
| ES_PASSWORD   | 0x20   | +/-Password. Displays a masking character in place of each character that is typed into the edit control, which conceals the text.                                                                                                                                                                                           |
| ES_READONLY   | 0x800  | +/-ReadOnly. Prevents the user from typing or editing text in the edit control.                                                                                                                                                                                                                                              |
| ES_RIGHT      | 0x2    | +/-Right. Right-aligns text in a multiline edit control.                                                                                                                                                                                                                                                                     |
| ES_UPPERCASE  | 0x8    | +/-Uppercase. Converts all characters to uppercase as they are typed into the edit control.                                                                                                                                                                                                                                  |
|               |        | +/-WantReturn. Specifies that a carriage                                                                                                                                                                                                                                                                                     |

|               |        |                                                                                                                                                                                                                                                                                                                |
|---------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ES_WANTRETURN | 0x1000 | return be inserted when the user presses the ENTER key while typing text into a multiline edit control in a dialog box. If you do not specify this style, pressing the ENTER key has the same effect as pressing the dialog box's default push button. This style has no effect on a single-line edit control. |
|---------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



| UpDown styles   | Value | Description                                                                                                                                                                                                                      |
|-----------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default         |       | UDS_ARROWKEYS, UDS_ALIGNRIGHT, UDS_SETBUDDYINT, and UDS_AUTOBUDDY.                                                                                                                                                               |
| forced          |       | None                                                                                                                                                                                                                             |
| UDS_WRAP        | 0x1   | Named option "Wrap". Causes the control to wrap around to the other end of its range when the user attempts to go beyond the minimum or maximum. Without <i>Wrap</i> , the control stops when the minimum or maximum is reached. |
| UDS_SETBUDDYINT | 0x2   | Causes the UpDown control to set the text of the buddy control (using the WM_SETTEXT message) when the position changes. However, if the buddy is a ListBox, the ListBox's current selection is changed instead.                 |
| UDS_ALIGNRIGHT  | 0x4   | Named option "Right" (default). Positions UpDown on the right side of its buddy control.                                                                                                                                         |
| UDS_ALIGNLEFT   | 0x8   | Named option "Left". Positions UpDown on the left side of its buddy control.                                                                                                                                                     |

|                 |       |                                                                                                                                                                                                                                                                                                                                |
|-----------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UDS_AUTOBUDDY   | 0x10  | Automatically selects the previous control in the z-order as the UpDown control's buddy control.                                                                                                                                                                                                                               |
| UDS_ARROWKEYS   | 0x20  | Allows the user to press the Up or Down arrow keys on the keyboard to increase or decrease the UpDown control's position.                                                                                                                                                                                                      |
| UDS_HORZ        | 0x40  | Named option "Horz". Causes the control's arrows to point left and right instead of up and down.                                                                                                                                                                                                                               |
| UDS_NOTHOUSANDS | 0x80  | Does not insert a thousands separator between every three decimal digits in the buddy control.                                                                                                                                                                                                                                 |
| UDS_HOTTRACK    | 0x100 | Causes the control to exhibit "hot tracking" behavior. That is, it highlights the control's buttons as the mouse passes over them. The flag is ignored on Windows XP when the desktop theme overrides it.                                                                                                                      |
| (hex)           |       | <p>The number format displayed inside the buddy control may be changed from decimal to hexadecimal by following this example:</p> <pre>Gui.Opt("+LastFound") SendMessage(1133, 16, 0, "msctls_updown321") ; 1133 is UDM_SETBASE</pre> <p>However, this affects only the buddy control, not the UpDown's reported position.</p> |

| Picture styles     | Value | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default            |       | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| forced             |       | SS_ICON for icons and cursors.<br>SS_BITMAP for other image types.                                                                                                                                                                                                                                                                                                                                                                                                       |
| SS_REALSIZECONTROL | 0x40  | [Windows XP or later] Adjusts the bitmap to fit the size of the control.                                                                                                                                                                                                                                                                                                                                                                                                 |
| SS_CENTERIMAGE     | 0x200 | Centers the bitmap in the control. If the bitmap is too large, it will be clipped. For text controls If the control contains a single line of text, the text is centered vertically within the available height of the control<br><br>Microsoft Windows XP: This style bit no longer results in unused portions of the control being filled with the color of the top left pixel of the bitmap or icon. Unused portions of the control will remain the background color. |

| Button, Checkbox, Radio, and GroupBox styles | Value | Description                                                                                                                                                            |
|----------------------------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default                                      |       | BS_MULTILINE (except for GroupBox, and except for buttons, checkboxes, and radios that have no explicitly set width or height, nor any CR/LF characters in their text) |

|                             |        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |        | <a href="#">WS_TABSTOP</a> (except for GroupBox)<br>-- however, radio buttons other than the first of each radio group lack <a href="#">WS_TABSTOP</a> by default.                                                                                                                                                                                                                                                                                                                                |
| forced                      |        | Button: <a href="#">BS_PUSHBUTTON</a> or <a href="#">BS_DEFPUSHBUTTON</a><br>Radio: <a href="#">BS_AUTORADIOBUTTON</a><br>Checkbox: <a href="#">BS_AUTOCHECKBOX</a> or <a href="#">BS_AUTO3STATE</a><br>GroupBox: <a href="#">BS_GROUPBOX</a>                                                                                                                                                                                                                                                     |
| <a href="#">BS_NOTIFY</a>   | 0x4000 | Enables <a href="#">DoubleClick</a> , <a href="#">Focus</a> and <a href="#">LoseFocus</a> events. This style is added automatically by <a href="#">OnEvent</a> when any these events is registered, but is not removed automatically.<br><br>If set, the second click in a double-click has no effect other than to raise the event. If not set, a double-click acts like two single-clicks for the purpose of cycling checkmark values or causing the visual effect of the button being clicked. |
| <a href="#">BS_LEFT</a>     | 0x100  | +/-Left. Left-aligns the text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <a href="#">BS_PUSHLIKE</a> | 0x1000 | Makes a checkbox or radio button look and act like a push button. The button looks raised when it isn't pushed or checked, and sunken when it is pushed or checked.                                                                                                                                                                                                                                                                                                                               |
| <a href="#">BS_RIGHT</a>    | 0x200  | +/-Right. Right-aligns the text.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|                             |        | +Right (i.e. +Right includes both <a href="#">BS_RIGHT</a> and <a href="#">BS_RIGHTBUTTON</a> , but -Right removes only <a href="#">BS_RIGHT</a> ,                                                                                                                                                                                                                                                                                                                                                |

|                  |        |                                                                                                                                                                                                                                                                                                   |
|------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BS_RIGHTBUTTON   | 0x20   | not BS_RIGHTBUTTON). Positions a checkbox square or radio button circle on the right side of the control's available width instead of the left.                                                                                                                                                   |
| BS_BOTTOM        | 0x800  | Places the text at the bottom of the control's available height.                                                                                                                                                                                                                                  |
| BS_CENTER        | 0x300  | +/-Center. Centers the text horizontally within the control's available width.                                                                                                                                                                                                                    |
| BS_DEFPUSHBUTTON | 0x1    | +/-Default. Creates a push button with a heavy black border. If the button is in a dialog box, the user can select the button by pressing the ENTER key, even when the button does not have the input focus. This style is useful for enabling the user to quickly select the most likely option. |
| BS_MULTILINE     | 0x2000 | +/-Wrap. Wraps the text to multiple lines if the text is too long to fit on a single line in the control's available width. This also allows linefeed (\n) to start new lines of text.                                                                                                            |
| BS_TOP           | 0x400  | Places text at the top of the control's available height.                                                                                                                                                                                                                                         |
| BS_VCENTER       | 0xC00  | Vertically centers text in the control's available height.                                                                                                                                                                                                                                        |
| BS_FLAT          | 0x8000 | Specifies that the button is two-dimensional; it does not use the default shading to create a 3-D effect.                                                                                                                                                                                         |



| DropDownList and<br>ComboBox styles | Value | Description |
|-------------------------------------|-------|-------------|
|                                     |       |             |

|                      |        |                                                                                                                                                                                                                            |
|----------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default              |        | WS_TABSTOP (+/-Tabstop)<br>DropDownList: WS_VSCROLL<br>ComboBox: WS_VSCROLL,<br>CBS_AUTOHSCROLL                                                                                                                            |
| forced               |        | DropDownList:<br>CBS_DROPDOWNLIST<br>ComboBox: Either<br>CBS_DROPDOWN or<br>CBS_SIMPLE                                                                                                                                     |
| CBS_AUTOHSCROLL      | 0x40   | +/-Limit. Automatically scrolls the text in an edit control to the right when the user types a character at the end of the line. If this style is not set, only text that fits within the rectangular boundary is enabled. |
| CBS_DISABLENOSCROLL  | 0x800  | Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.                   |
| CBS_LOWERCASE        | 0x4000 | +/-Lowercase. Converts to lowercase any uppercase characters that are typed into the edit control of a combo box.                                                                                                          |
| CBS_NOINTEGRALHEIGHT | 0x400  | Specifies that the combo box will be exactly the size specified by the application when it created the combo box. Usually, Windows CE sizes a combo box so that it does not display partial items.                         |
|                      |        | Converts text typed in the combo                                                                                                                                                                                           |

|                |        |                                                                                                                                                                                                                                                         |
|----------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CBS_OEMCONVERT | 0x80   | box edit control from the Windows CE character set to the OEM character set and then back to the Windows CE set. This style is most useful for combo boxes that contain file names. It applies only to combo boxes created with the CBS_DROPDOWN style. |
| CBS_SIMPLE     | 0x1    | +/-Simple (ComboBox only). Displays the list box at all times. The current selection in the list box is displayed in the edit control.                                                                                                                  |
| CBS_SORT       | 0x100  | +/-Sort. Sorts the items in the drop-list alphabetically.                                                                                                                                                                                               |
| CBS_UPPERCASE  | 0x2000 | +/-Uppercase. Converts to uppercase any lowercase characters that are typed into the edit control of a ComboBox.                                                                                                                                        |



| <b>ListBox</b> styles | Value  | Description                                                                                                                            |
|-----------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------|
| default               |        | WS_TABSTOP, LBS_USETABSTOPS, WS_VSCROLL, and WS_EX_CLIENTEDGE (extended style E0x200).                                                 |
| forced                |        | LBS_NOTIFY (supports detection of double-clicks)                                                                                       |
| LBS_DISABLENOSCROLL   | 0x1000 | Shows a disabled vertical scroll bar for the list box when the box does not contain enough items to scroll. If you do not specify this |

|                      |        |                                                                                                                                                                                                                           |
|----------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |        | style, the scroll bar is hidden when the list box does not contain enough items.                                                                                                                                          |
| LBS_NOINTEGRALHEIGHT | 0x100  | Specifies that the list box will be exactly the size specified by the application when it created the list box.                                                                                                           |
| LBS_EXTENDEDSEL      | 0x800  | +/-Multi. Allows multiple selections via control-click and shift-click.                                                                                                                                                   |
| LBS_MULTIPLESEL      | 0x8    | A simplified version of multi-select in which control-click and shift-click are not necessary because normal left clicks serve to extend the selection or de-select a selected item.                                      |
| LBS_NOSEL            | 0x4000 | +/-ReadOnly. Specifies that the user can view list box strings but cannot select them.                                                                                                                                    |
| LBS_SORT             | 0x2    | +/-Sort. Sorts the items in the list box alphabetically.                                                                                                                                                                  |
| LBS_USETABSTOPS      | 0x80   | Enables a ListBox to recognize and expand tab characters when drawing its strings. The default tab positions are 32 dialog box units. A dialog box unit is equal to one-fourth of the current dialog box base-width unit. |



| ListView styles | Value | Description                            |
|-----------------|-------|----------------------------------------|
|                 |       | WS_TABSTOP, LVS_RE<br>LVS_SHOWSELALWAY |

|                    |        |                                                                                                                                                                   |
|--------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default            |        | <a href="#">LVS_EX_FULLROWSELECT</a><br><a href="#">LVS_EX_HEADERDRAGINIT</a><br><a href="#">LVS_EX_CLIENTEDGE</a>                                                |
| forced             |        | None                                                                                                                                                              |
| LVS_ALIGNLEFT      | 0x800  | Items are left-aligned in icon and small icon view.                                                                                                               |
| LVS_ALIGNTOP       | 0      | Items are aligned with the top of the list-view control in icon and small icon view. This is the default.                                                         |
| LVS_AUTOARRANGE    | 0x100  | Icons are automatically kept arranged in icon and small icon view.                                                                                                |
| LVS_EDITLABELS     | 0x200  | +/-ReadOnly. Specifying +ReadOnly (or +0x200) allows the user to edit the first field of a row in place.                                                          |
| LVS_ICON           | 0      | +Icon. Specifies large-icon view.                                                                                                                                 |
| LVS_LIST           | 0x3    | +List. Specifies list view.                                                                                                                                       |
| LVS_NOCOLUMNHEADER | 0x4000 | +/-Hdr. Avoids displaying column headers in report view.                                                                                                          |
| LVS_NOLABELWRAP    | 0x80   | Item text is displayed on a single line in icon view. By default, text may wrap in icon view.                                                                     |
| LVS_NOSCROLL       | 0x2000 | Scrolling is disabled. All items must be within the client area. This style is not compatible with <a href="#">LVS_LIST</a> or <a href="#">LVS_REPORT</a> styles. |
| LVS NOSORTHEADER   | 0x8000 | +/-NoSortHdr. Column headers do not work like buttons. This style can be used if clicking a column header does not sort the list.                                 |

|                     |        |                                                                                                                                                     |
|---------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |        | header in report view does not carry out an action, such as                                                                                         |
| LVS_OWNERDATA       | 0x1000 | This style specifies a virtual view control (not directly supported by AutoHotkey).                                                                 |
| LVS_OWNERDRAWFIXED  | 0x400  | The owner window can paint in report view in response to WM_DRAWITEM message (not directly supported by AutoHotkey).                                |
| LVS_REPORT          | 0x1    | +Report. Specifies report view.                                                                                                                     |
| LVS_SHAREIMAGELISTS | 0x40   | The image list will not be destroyed when the control is destroyed. This style enables the use of the image lists with multiple list view controls. |
| LVS_SHOWSELALWAYS   | 0x8    | The selection, if any, is always shown, even if the control does not have keyboard focus.                                                           |
| LVS_SINGLESEL       | 0x4    | +/-Multi. Only one item at a time can be selected. By default, multiple items can be selected.                                                      |
| LVS_SMALLICON       | 0x2    | +IconSmall. Specifies small icon view.                                                                                                              |
| LVS_SORTASCENDING   | 0x10   | +/-Sort. Rows are sorted in ascending order based on the contents of the first field.                                                               |
| LVS_SORTDESCENDING  | 0x20   | +/-SortDesc. Same as above but in descending order.                                                                                                 |

**Extended ListView styles** require the [LV prefix](#) when used with the Gui methods/properties. Some extended styles introduced in Windows XP or later versions are not listed here. For a full list, see [MSDN: Extended List-View Styles](#).

|                     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LVS_EX_BORDERSELECT | LV0x8000  | When an item is selected the border color of the item changes rather than the item being highlighted (might be non-functional in recent operating systems).                                                                                                                                                                                                                                                                                                                                                                     |
| LVS_EX_CHECKBOXES   | LV0x4     | <p>+/--Checked. Displays a checkbox with each item. When set to this style, the control creates a state image list with two images using DrawFrameControl. image 1 is the unchecked state image 2 is the checked state. Setting the state image to 0 removes the check box altogether.</p> <p>Windows XP or later: Checkboxes are visible and functional in all list-view modes except the icon view mode introduced in Windows XP. Clicking a checkbox in icon view mode only selects the item; the state does not change.</p> |
| LVS_EX_DOUBLEBUFFER | LV0x10000 | Windows XP or later: Enables double-buffering, which reduces flicker. This extended style also enables alpha-blended mouse selection on systems where it is supported.                                                                                                                                                                                                                                                                                                                                                          |
| LVS_EX_FLATSB       | LV0x100   | Enables flat scroll bars in list view.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|                     |           | When a row is selected, all                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                       |          |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LVS_EX_FULLROWSELECT  | LV0x20   | fields are highlighted. This style is available only in conjunction with the <a href="#">LVS_REPORT</a> style.                                                                                                                                                                                                                                                                                                    |
| LVS_EX_GRIDLINES      | LV0x1    | +/-Grid. Displays gridlines between rows and columns. This style is available only in conjunction with the <a href="#">LVS_REPORT</a> style.                                                                                                                                                                                                                                                                      |
| LVS_EX_HEADERDRAGDROP | LV0x10   | Enables drag-and-drop reordering of columns in a list-view control. This style is only available for list-view controls that use the <a href="#">LVS_REPORT</a> style.                                                                                                                                                                                                                                            |
| LVS_EX_INFOTIP        | LV0x400  | When a list-view control uses the <a href="#">LVS_EX_INFOTIP</a> style, the <a href="#">LVN_GETINFOTIP</a> notification message is sent to the parent window before displaying an item's ToolTip.                                                                                                                                                                                                                 |
| LVS_EX_LABELTIP       | LV0x4000 | If a partially hidden label in list-view mode lacks a ToolTip, the list-view control will use the label. If this style is not set, the list-view control will use the hidden labels only for the icon mode. Requires Windows XP or later, or the DLLs distributed with Internet Explorer 5.0. Note: On some versions of Windows, this style might not work properly if the GUI thread is set to be always-on-top. |
|                       |          | If the list-view control has the <a href="#">LVS_AUTOARRANGE</a> style, the control will not autoarrange icons until one or more windows are defined (see                                                                                                                                                                                                                                                         |

|                         |            |                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LVS_EX_MULTITWORKAREAS  | LV0x2000   | LVM_SETWORKAREAS effective, this style must be set before any work areas are created and any items have been added to the control.                                                                                                                                                                                                                          |
| LVS_EX_ONECLICKACTIVATE | LV0x40     | The list-view control sends the LVN_ITEMACTIVATE notification message to the parent window when the user clicks an item. This style also enables hot tracking in the list-view control. Hot tracking means that when the mouse cursor moves over an item, the item is highlighted but not selected.                                                         |
| LVS_EX_REGIONAL         | LV0x200    | Sets the list-view window to include only the item icons and text using SetWindowRgn to define an area that is not part of an icon area excluded from the window. This style is only available for list-view controls that use the LVS_ICON style.                                                                                                          |
| LVS_EX_SIMPLESELECT     | LV0x100000 | In icon view, moves the status bar image of the control to the right of the large icon rendering. In other views other than icon view, there is no change. When the user clicks an item, the state of the control changes by using the spacebar to cycle over selected items, and the focus moves to the item with the focus. Requires Windows XP or later. |
| LVS_EX_SUBITEMIMAGES    | LV0x2      | Allows images to be displayed in subitem fields beyond the first. This style is available only in conjunction with the LVS_REPORT style.                                                                                                                                                                                                                    |
|                         |            |                                                                                                                                                                                                                                                                                                                                                             |

|                         |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LVS_EX_TRACKSELECT      | LV0x8    | Enables hot-track selection in a list-view control. Hot track selection means that an item is automatically selected when the cursor remains over the item for a certain period of time. The timeout can be changed from the command line with a system setting with a <code>LVM_SETHOVERTIME</code> registry value. This style applies to all styles in a list-view control. You can enable or disable whether hot-track selection is enabled by calling <code>SystemParametersInfo</code> . |
| LVS_EX_TWOCLICKACTIVATE | LV0x80   | The list-view control sends <code>LVN_ITEMACTIVATE</code> notification message to the parent window when the user double-clicks an item. This style enables hot tracking in the list-view control. Hot tracking means that when the cursor moves over an item, it is highlighted by a background color. If selected, the item is highlighted by a background color.                                                                                                                           |
| LVS_EX_UNDERLINECOLD    | LV0x1000 | Causes those non-hot items that are activated to be displayed with underlined text. This style requires that <a href="#">LVS_EX_TWOCLICKACTIVATE</a> be set also.                                                                                                                                                                                                                                                                                                                             |
| LVS_EX_UNDERLINEHOT     | LV0x800  | Causes those hot items that are activated to be displayed with underlined text. This style requires that <a href="#">LVS_EX_ONECLICKACTIVATE</a> or <a href="#">LVS_EX_TWOCLICKACTIVATE</a> be set also.                                                                                                                                                                                                                                                                                      |

also be set.

| TreeView styles     | Value  | Description                                                                                                                                                                                                                             |
|---------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default             |        | WS_TABSTOP,<br>TVS_SHOWSELALWAYS,<br>TVS_HASLINES,<br>TVS_LINESATROOT,<br>TVS_HASBUTTONS,<br>WS_EX_CLIENTEDGE<br>(E0x200)                                                                                                               |
| forced              |        | None                                                                                                                                                                                                                                    |
| TVS_CHECKBOXES      | 0x100  | +/-Checked. Displays a checkbox next to each item.                                                                                                                                                                                      |
| TVS_DISABLEDRAHDROP | 0x10   | Prevents the tree-view control from sending TVN_BEGINDRAG notification messages.                                                                                                                                                        |
| TVS_EDITLABELS      | 0x8    | +/-ReadOnly. Allows the user to edit the names of tree-view items.                                                                                                                                                                      |
| TVS_FULLROWSELECT   | 0x1000 | Enables full-row selection in the tree view. The entire row of the selected item is highlighted, and clicking anywhere on an item's row causes it to be selected. This style cannot be used in conjunction with the TVS_HASLINES style. |
| TVS_HASBUTTONS      | 0x1    | +/-Buttons. Displays plus (+) and minus (-) buttons next to parent items. The user clicks the buttons to expand or collapse a parent item's list of child items. To include buttons with items at the                                   |

|                   |        |                                                                                                                                                       |
|-------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |        | root of the tree view, <a href="#">TVS_LINESATROOT</a> must also be specified.                                                                        |
| TVS_HASLINES      | 0x2    | +/-Lines. Uses lines to show the hierarchy of items.                                                                                                  |
| TVS_INFOTIP       | 0x800  | Obtains ToolTip information by sending the TVN_GETINFOTIP notification.                                                                               |
| TVS_LINESATROOT   | 0x4    | +/-Lines. Uses lines to link items at the root of the tree-view control. This value is ignored if <a href="#">TVS_HASLINES</a> is not also specified. |
| TVS_NOHSCROLL     | 0x8000 | +/-HScroll. Disables horizontal scrolling in the control. The control will not display any horizontal scroll bars.                                    |
| TVS_NONEVENHEIGHT | 0x4000 | Sets the height of the items to an odd height with the TVM_SETITEMHEIGHT message. By default, the height of items must be an even value.              |
| TVS_NOSCROLL      | 0x2000 | Disables both horizontal and vertical scrolling in the control. The control will not display any scroll bars.                                         |
| TVS_NOTOOLTIPS    | 0x80   | Disables ToolTips.                                                                                                                                    |
| TVS_RTLREADING    | 0x40   | Causes text to be displayed from right-to-left (RTL). Usually, windows display text left-to-right (LTR).                                              |
| TVS_SHOWSELALWAYS | 0x20   | Causes a selected item to remain selected when the tree-view control loses focus.                                                                     |

|                  |       |                                                                                                                                                                                                                                   |
|------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TVS_SINGLEEXPAND | 0x400 | Causes the item being selected to expand and the item being unselected to collapse upon selection in the tree-view. If the user holds down the CTRL key while selecting an item, the item being unselected will not be collapsed. |
| TVS_TRACKSELECT  | 0x200 | Enables hot tracking of the mouse in a tree-view control.                                                                                                                                                                         |



| <a href="#">Date Time</a> styles | Value | Description                                                                                                                                                                                                                 |
|----------------------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default                          |       | <a href="#">DTS_SHORTDATECENT</a> and <a href="#">WS_TABSTOP</a> (+/-Tab)                                                                                                                                                   |
| forced                           |       | None                                                                                                                                                                                                                        |
| DTS_UPDOWN                       | 0x1   | Provides an up-down control of the control to modify date which replaces the of the month calendar that would be available.                                                                                                 |
| DTS_SHOWNONE                     | 0x2   | Displays a checkbox inside that users can uncheck to make control have no date/time selection. Whenever the control has <a href="#">Gui.Submit</a> and <a href="#">GuiCtrl.Value</a> retrieve a blank value (empty string). |
| DTS_SHORTDATEFORMAT              | 0x0   | Displays the date in short format. In newer locales, it looks like 6/1/05. On older operating systems, the year might be displayed. The <a href="#">DTS_SHORTDATECENT</a> is the default and not                            |

|                            |      |                                                                                                                                                                                                                                                                                                            |
|----------------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            |      | DTS_SHORTDATEFORM                                                                                                                                                                                                                                                                                          |
| DTS_LONGDATEFORMAT         | 0x4  | Format option "LongDate". date in long format. In some locales, it looks like Wednesday, June 1, 2005.                                                                                                                                                                                                     |
| DTS_SHORTDATECENTURYFORMAT | 0xC  | Format option blank/omit the century. The date in short format without the year. In some locales, it looks like 6/1/2005. If the system's version of Comctl32.dll is older than 6.0.2600.5512, this option is not supported and DTS_SHORTDATEFORMAT is automatically substituted.                          |
| DTS_TIMEFORMAT             | 0x9  | Format option "Time". Displays the time, which in some locales looks like 5:31:42 PM.                                                                                                                                                                                                                      |
| DTS_APPCANPARSE            | 0x10 | Not yet supported. Allows the calendar to parse user input and take action. It enables users to click on the client area of the control with the mouse or press the F2 key. The control sends DTN_USERSTRING notification messages when users are finished with their input.                               |
| DTS_RIGHTALIGN             | 0x20 | +/-Right. The calendar will align the right side of the control to the right side of the window or left.                                                                                                                                                                                                   |
|                            |      | The colors of the day numbers in the drop-down calendar obey the <code>Gui.SetFont</code> or the <code>c (Color)</code> property. To change the colors of other controls in the calendar, follow this example:<br><pre>Gui.Opt("+LastSendDlgItemControl", "SysDateTimePick", "Color", 0x4, 0xFFAA99,</pre> |

(colors inside the drop-down calendar)

```
; 0x1006 is
DTM_SETMCCOLOR
MCSC_MONTHBK
(background color)
The color must
be specified in B
R, G, B format (re
blue component
swapped).
```



| MonthCal styles | Value | Description                                                                                                                                                                                                                                                                                                                                       |
|-----------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default         |       | WS_TABSTOP (only on Windows Vista and later)                                                                                                                                                                                                                                                                                                      |
| forced          |       | None                                                                                                                                                                                                                                                                                                                                              |
| MCS_DAYSTATE    | 0x1   | Makes the control send MCN_GETDAYSTATE notifications to request information about which days should be displayed in bold. [Not yet supported]                                                                                                                                                                                                     |
| MCS_MULTISELECT | 0x2   | Named option "Multi". Allows the user to select a range of dates rather than being limited to a single date. By default, the maximum range is 366 days, which can be changed by sending the MCM_SETMAXSELCOUNT message to the control. For example:<br><pre>Gui.Opt(+LastFound)<br/>SendMessage(0x1004,<br/>7, 0,<br/>"SysMonthCal321") ; 7</pre> |

|                   |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |      | <p>days. 0x1004 is MCM_SETMAXSELCOUNT.</p>                                                                                                                                                                                                                                                                                                                                                                                                                       |
| MCS_WEEKNUMBERS   | 0x4  | Displays week numbers (1-52) to the left of each row of days. Week 1 is defined as the first week that contains at least four days.                                                                                                                                                                                                                                                                                                                              |
| MCS_NOTODAYCIRCLE | 0x8  | Prevents the circling of today's date within the control.                                                                                                                                                                                                                                                                                                                                                                                                        |
| MCS_NOTODAY       | 0x10 | Prevents the display of today's date at the bottom of the control.                                                                                                                                                                                                                                                                                                                                                                                               |
| (colors)          |      | <p>The colors of the day numbers inside the calendar obey that set by <code>Gui.SetFont</code> or the <code>c (Color)</code> option. To change the colors of other parts of the calendar, follow this example:</p> <pre>Gui.Opt("+LastFound") SendMessage(0x100A, 5, 0xFFAA99, "SysMonthCal321") ; 0x100A is MCM_SETCOLOR. 5 is MCSC_TITLETEXT (color of title text). The color must be specified in BGR vs. RGB format (red and blue components swapped).</pre> |

| Slider styles      | Value | Description                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default            |       | WS_TABSTOP (+/-Tabstop)                                                                                                                                                                                                                                                                                                                                                                                                                       |
| forced             |       | None                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| TBS_VERT           | 0x2   | +/-Vertical. The control is oriented vertically.                                                                                                                                                                                                                                                                                                                                                                                              |
| TBS_LEFT           | 0x4   | +/-Left. The control displays tick marks at the top of the control (or to its left if TBS_VERT is present). Same as TBS_TOP                                                                                                                                                                                                                                                                                                                   |
| TBS_TOP            | 0x4   | same as TBS_LEFT.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| TBS_BOTH           | 0x8   | +/-Center. The control displays tick marks on both sides of the control. This will be both top and bottom when used with TBS_HORZ or both left and right if used with TBS_VERT.                                                                                                                                                                                                                                                               |
| TBS_AUTOTICKS      | 0x1   | The control has a tick mark for each increment in its range of values. Use +/-TickInterval to have more flexibility.                                                                                                                                                                                                                                                                                                                          |
| TBS_ENABLESELRANGE | 0x20  | <p>The control displays a selection range only. The tick marks at the starting and ending positions of a selection range are displayed as triangles (instead of vertical dashes), and the selection range is highlighted (highlighting might require that the theme be removed via <code>Gui.Opt("theme")</code>).</p> <p>To set the selection range, follow this example, which sets the starting position to 55 and the ending position</p> |

|                 |       |                                                                                                                                                                                                                                                                             |
|-----------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |       | <p>to 66:</p> <pre>SendMessage, 1035, 1, 55, msctls_trackbar321, WinTitle SendMessage, 1036, 1, 66, msctls_trackbar321, WinTitle</pre>                                                                                                                                      |
| TBS_FIXEDLENGTH | 0x40  | +/-Thick. Allows the thumb's size to be changed.                                                                                                                                                                                                                            |
| TBS_NOTHUMB     | 0x80  | The control does not display the moveable bar.                                                                                                                                                                                                                              |
| TBS_NOTICKS     | 0x10  | +/-NoTicks. The control does not display any tick marks.                                                                                                                                                                                                                    |
| TBS_TOOLTIPS    | 0x100 | +/-Tooltip. The control supports ToolTips. When a control is created using this style, it automatically creates a default Tooltip control that displays the slider's current position. You can change where the ToolTips are displayed by using the TBM_SETTIPSIDE message. |
| TBS_REVERSED    | 0x200 | Unfortunately, this style has no effect on the actual behavior of the control, so there is probably no point in using it (instead, use +Invert in the control's options to reverse it). Depending on OS version, this style might require Internet Explorer 5.0 or greater. |
|                 |       | Unfortunately, this style has no effect on the actual behavior of the control,                                                                                                                                                                                              |

|                |       |                                                                                                                                 |
|----------------|-------|---------------------------------------------------------------------------------------------------------------------------------|
| TBS_DOWNISLEFT | 0x400 | so there is probably no point in using it. Depending on OS version, this style might require Internet Explorer 5.01 or greater. |
|----------------|-------|---------------------------------------------------------------------------------------------------------------------------------|

| Progress styles | Value | Description                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default         |       | None                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| forced          |       | None                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| PBS_SMOOTH      | 0x1   | +/-Smooth. The progress bar displays progress status in a smooth scrolling bar instead of the default segmented bar. When this style is present, the control automatically reverts to the Classic Theme appearance on Windows XP or later.                                                                                                                                                                                                            |
| PBS_VERTICAL    | 0x4   | +/-Vertical. The progress bar displays progress status vertically, from bottom to top.                                                                                                                                                                                                                                                                                                                                                                |
| PBS_MARQUEE     | 0x8   | [Requires Windows XP or later] The progress bar moves like a marquee; that is, each change to its position causes the bar to slide further along its available length until it wraps around to the other side. A bar with this style has no defined position. Each attempt to change its position will instead slide the bar by one increment.<br><br>This style is typically used to indicate an ongoing operation whose completion time is unknown. |

| Tab styles         | Value | Description                                                                                                                                                                                       |
|--------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default            |       | WS_TABSTOP and TCS_MULTILINE                                                                                                                                                                      |
| forced             |       | WS_CLIPSIBLINGS<br>TCS_OWNERDRAWFIXED:<br>forced on or off as required by the control's background color and/or text color.                                                                       |
| TCS_SCROLLOPPOSITE | 0x1   | Unneeded tabs scroll to the opposite side of the control when a tab is selected.                                                                                                                  |
| TCS_BOTTOM         | 0x2   | +/-Bottom. Tabs appear at the bottom of the control instead of the top.                                                                                                                           |
| TCS_RIGHT          | 0x2   | Tabs appear vertically on the right side of controls that use the TCS_VERTICAL style.                                                                                                             |
| TCS_MULTISELECT    | 0x4   | Multiple tabs can be selected by holding down CTRL when clicking. This style must be used with the TCS_BUTTONS style.                                                                             |
| TCS_FLATBUTTONS    | 0x8   | Selected tabs appear as being indented into the background while other tabs appear as being on the same plane as the background. This style only affects tab controls with the TCS_BUTTONS style. |
| TCS_FORCEICONLEFT  | 0x10  | Icons are aligned with the left edge of each fixed-width tab. This style can only be used with the TCS_FIXEDWIDTH style.                                                                          |

|                    |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TCS_FORCELABELLEFT | 0x20  | <p>Labels are aligned with the left edge of each fixed-width tab; that is, the label is displayed immediately to the right of the icon instead of being centered.</p> <p>This style can only be used with the <a href="#">TCS_FIXEDWIDTH</a> style, and it implies the <a href="#">TCS_FORCEICONLEFT</a> style.</p>                                                                                                                                                                                    |
| TCS_HOTTRACK       | 0x40  | Items under the pointer are automatically highlighted                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| TCS_VERTICAL       | 0x80  | <p>+/-Left or +/-Right. Tabs appear at the left side of the control, with tab text displayed vertically. This style is valid only when used with the <a href="#">TCS_MULTILINE</a> style. To make tabs appear on the right side of the control, also use the <a href="#">TCS_RIGHT</a> style.</p> <p>This style will not correctly display the tabs if a custom background color or text color is in effect. To workaround this, specify -Background and/or cDefault in the tab control's options.</p> |
| TCS_BUTTONS        | 0x100 | +/-Buttons. Tabs appear as buttons, and no border is                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|                       |        |                                                                                                                                                                                                                                                  |
|-----------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       |        | drawn around the display area.                                                                                                                                                                                                                   |
| TCS_SINGLELINE        | 0      | +/-Wrap. Only one row of tabs is displayed. The user can scroll to see more tabs, if necessary. This style is the default.                                                                                                                       |
| TCS_MULTILINE         | 0x200  | +/-Wrap. Multiple rows of tabs are displayed, if necessary, so all tabs are visible at once.                                                                                                                                                     |
| TCS_RIGHTJUSTIFY      | 0      | This is the default. The width of each tab is increased, if necessary, so that each row of tabs fills the entire width of the tab control.<br><br>This window style is ignored unless the <a href="#">TCS_MULTILINE</a> style is also specified. |
| TCS_FIXEDWIDTH        | 0x400  | All tabs are the same width. This style cannot be combined with the <a href="#">TCS_RIGHTJUSTIFY</a> style.                                                                                                                                      |
| TCS_RAGGEDRIGHT       | 0x800  | Rows of tabs will not be stretched to fill the entire width of the control. This style is the default.                                                                                                                                           |
| TCS_FOCUSONBUTTONDOWN | 0x1000 | The tab control receives the input focus when clicked.                                                                                                                                                                                           |
| TCS_OWNERDRAWFIXED    | 0x2000 | The parent window is responsible for drawing tabs.                                                                                                                                                                                               |
| TCS_TOOLTIPS          | 0x4000 | The tab control has a tooltip                                                                                                                                                                                                                    |

|                |        |                                                                |
|----------------|--------|----------------------------------------------------------------|
|                |        | control associated with it.                                    |
| TCS_FOCUSNEVER | 0x8000 | The tab control does not receive the input focus when clicked. |

| StatusBar styles | Value | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default          |       | SBARS_TOOLTIPS and SBARS_SIZEGRIP (the latter only if the window is resizable).                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| forced           |       | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| SBARS_TOOLTIPS   | 0x800 | <p>Displays a tooltip when the mouse hovers over a part of the status bar that: 1) has too much text to be fully displayed; or 2) has an icon but no text.</p> <p>The text of the ToolTip can be set via:</p> <pre>Gui.Opt("+LastFound") SendMessage(0x410, <b>0</b>, "Text to display", "mctls_statusbar321")</pre> <p>The bold <b>0</b> above is the zero-based part number. To use a part other than the first, specify 1 for second, 2 for the third, etc.<br/>NOTE: The ToolTip might never appear on certain OS versions.</p> |
| SBARS_SIZEGRIP   | 0x100 | Includes a sizing grip at the right end of the status bar. A sizing grip is similar to a sizing border; it is a rectangular area that the user can click and drag to resize the parent                                                                                                                                                                                                                                                                                                                                              |

window.

# Image Handles

To use an icon or bitmap handle in place of an image filename, use the following syntax:

```
HBITMAP:bitmap-handle
HICON:icon-handle
```

Replace *bitmap-handle* or *icon-handle* with the actual handle value. For example, `hicon:%handle%` (or `"hicon:" handle` in an [expression](#)), where *handle* is a variable containing an icon handle.

The following things support this syntax:

- `Gui.AddPicture` (and `GuiCtrl.Value` when setting a Picture control's content).
- `IL_Add`
- `LoadPicture`
- `SB.SetIcon`
- `ImageSearch`
- `Menu Tray, Icon` or `Menu MenuName, Icon`

A bitmap or icon handle is a numeric value which identifies a bitmap or icon in memory. The majority of scripts never need to deal with handles, as in most cases AutoHotkey takes care of loading the image from file and freeing it when it is no longer needed. The syntax shown above is intended for use when the script obtains an icon or bitmap handle from another source, such as by sending

the WM\_GETICON message to a window. It can also be used in combination with [LoadPicture](#) to avoid loading an image from file multiple times.

By default, AutoHotkey treats the handle as though it loaded the image from file - for example, a bitmap used on a Picture control is deleted when the GUI is destroyed, and an image will generally be deleted immediately if it needs to be resized. To avoid this, put an asterisk between the colon and handle. For example: `hbitmap: *%handle%` (or `"hbitmap: *" handle` in an expression). With the exception of ImageSearch, this forces the command to take a copy of the image.

## Examples

```
; Show a menu of the first n files matching a pattern, and their icons.
```

```
pattern := "%A_ScriptDir%*"
n := 15
```

```
; Allocate memory for a SHFILEINFOW struct.
```

```
VarSetCapacity(fileinfo, fsize := A_PtrSize + 688)
```

```
Loop, Files, %pattern%, FD
{
```

```
; Add a menu item for each file.
```

```
Menu F, Add, %A_LoopFileName%, donothing
```

```
; Get the file's icon.
```

```
if DllCall("shell32\SHGetFileInfow", "wstr",
A_LoopFileFullPath
, "uint", 0, "ptr", &fileinfo, "uint",
fsize, "uint", 0x100)
```

```
{
 hicon := NumGet(fileinfo, 0, "ptr")
 ; Set the menu item's icon.
 Menu F, Icon, %A_Index%&, HICON:%hicon%
 ; Because we used ":" and not ":", the
 icon will be automatically
 ; freed when the program exits or if the
 menu or item is deleted.
}
}
until A_Index = n
Menu F, Show
do nothing:
return
```

See also: [LoadPicture](#).

# Color names and RGB values

|                                                                                     | Color name | RGB value |
|-------------------------------------------------------------------------------------|------------|-----------|
|    | Black      | 000000    |
|    | Silver     | C0C0C0    |
|    | Gray       | 808080    |
|    | White      | FFFFFF    |
|    | Maroon     | 800000    |
|    | Red        | FF0000    |
|    | Purple     | 800080    |
|    | Fuchsia    | FF00FF    |
|   | Green      | 008000    |
|  | Lime       | 00FF00    |
|  | Olive      | 808000    |
|  | Yellow     | FFFF00    |
|  | Navy       | 000080    |
|  | Blue       | 0000FF    |
|  | Teal       | 008080    |
|  | Aqua       | 00FFFF    |

# OnNotify

Registers a function or method to be called when a control notification is received via the `WM_NOTIFY` message.

```
GuiControl.OnNotify(NotifyCode, Callback, AddRemove := 1)
```

## Parameters

### GuiControl

The `GuiControl` object of the control to monitor.

### NotifyCode

The control-defined notification code to monitor.

### Callback

The function, method or object to call when the event is raised.

If this parameter is a string, its meaning depends on whether the GUI has an `event sink` (that is, whether `GuiCreate`'s `EventObj` parameter was specified). If the GUI has an event sink, the string must be the name of a method belonging to the event sink; otherwise, it must be the name of a function.

To register a function regardless of whether the GUI has an event sink, pass a `function reference`.

## AddRemove

One of the following values:

**1** (the default): Call the callback after any previously registered callbacks.

**-1**: Call the callback before any previously registered callbacks.

**0**: Do not call the callback.

## WM\_NOTIFY

Certain types of controls send a `WM_NOTIFY` message whenever an interesting event occurs or the control requires information from the program. The *lParam* parameter of this message contains a pointer to a structure containing information about the notification. The type of structure depends on the notification code and the type of control which raised the notification, but is always based on `NMHDR`.

To determine which notifications are available (if any), what type of structure they provide and how they interpret the return value, refer to the control's documentation. [Control Library \(MSDN\)](#) contains links to each of the the Windows common controls. The notification codes (numbers) can be found in the Windows SDK, or by searching the Internet.

AutoHotkey uses the *idFrom* and *hwndFrom* fields to identify which control sent the notification, in order to dispatch it to the appropriate object. The *code* field contains the notification code. Since these must match up to the *GuiControl* and *NotifyCode* used to register the callback, they are rarely useful to the script.

## Callback Parameters

The [notes for OnEvent](#) regarding `this` and bound functions also apply to `OnNotify`.

The callback receives two parameters:

```
Callback(GuiControl, lParam)
```

As noted above, *lParam* is the address of a notification structure derived from `NMHDR`. Its exact type depends on the type of control and notification code. If the structure contains additional information about the notification, the callback can retrieve it with `NumGet` and/or `StrGet`.

## Callback Return Value

If multiple callbacks have been registered for an event, a callback may return a non-empty value to prevent any remaining callbacks from being called.

The return value may have additional meaning, depending on the notification. For example, the ListView notification `LVN_BEGINLABELEDIT` (-175 or -105) prevents the user from editing the label if the callback returns `TRUE` (1).

## Related

These notes for [OnEvent](#) also apply to [OnNotify: Threads](#), [Destroying the GUI](#).

[OnCommand](#) can be used for notifications which are sent as a `WM_COMMAND` message.

# OnCommand

Registers a function or method to be called when a control notification is received via the `WM_COMMAND` message.

```
GuiControl.OnCommand(NotifyCode, Callback ,
AddRemove := 1)
```

## Parameters

### GuiControl

The `GuiControl` object of the control to monitor.

### NotifyCode

The control-defined notification code to monitor.

### Callback

The function, method or object to call when the event is raised.

If this parameter is a string, its meaning depends on whether the GUI has an `event sink` (that is, whether `GuiCreate`'s `EventObj` parameter was specified). If the GUI has an event sink, the string must be the name of a method belonging to the event sink; otherwise, it must be the name of a function.

To register a function regardless of whether the GUI has an event sink, pass a `function reference`.

## AddRemove

One of the following values:

**1** (the default): Call the callback after any previously registered callbacks.

**-1**: Call the callback before any previously registered callbacks.

**0**: Do not call the callback.

## WM\_COMMAND

Certain types of controls send a [WM\\_COMMAND](#) message whenever an interesting event occurs. These are usually standard Windows controls which have been around a long time, as newer controls use the [WM\\_NOTIFY](#) message (see [OnNotify](#)). Commonly used notification codes are translated to events, which the script can monitor with [OnEvent](#).

The message's parameters contain the control ID, HWND and notification code, which AutoHotkey uses to dispatch the notification to the appropriate callback. There are no additional parameters.

To determine which notifications are available (if any), refer to the control's documentation. [Control Library \(MSDN\)](#) contains links to each of the the Windows common controls (however, only a few of these use [WM\\_COMMAND](#)). The notification codes (numbers) can be found in the Windows SDK, or by searching the Internet.

## Callback Parameters

The [notes for OnEvent](#) regarding `this` and bound functions also apply to `OnCommand`.

The callback receives one parameter:

```
Callback(GuiControl)
```

## Callback Return Value

If multiple callbacks have been registered for an event, a callback may return a non-empty value to prevent any remaining callbacks from being called.

The return value is ignored by the control.

## Related

These notes for [OnEvent](#) also apply to [OnCommand: Threads](#), [Destroying the GUI](#).

[OnNotify](#) can be used for notifications which are sent as a `WM_NOTIFY` message.

# Standard Windows Fonts

For use with [Gui.SetFont](#).

Recommend Fonts are highlighted in yellow.

| FONT NAME                       | Win95 | WinNT | Win98 | Win2000 | WinMe | WinXP |
|---------------------------------|-------|-------|-------|---------|-------|-------|
| Abadi MT<br>Condensed<br>Light  |       |       | x     |         |       |       |
| Arial                           | x     | x     | x     | x       | x     | x     |
| Arial<br>Alternative<br>Regular |       |       |       |         | x     |       |
| Arial<br>Alternative<br>Symbol  |       |       |       |         | x     |       |
| Arial Black                     |       |       | x     | x       | x     | x     |
| Arial Bold                      | x     | x     | x     | x       | x     | x     |
| Arial Bold<br>Italic            | x     | x     | x     | x       | x     | x     |
| Arial Italic                    | x     | x     | x     | x       | x     | x     |
| Book Antiqua                    |       |       | x     |         |       |       |
| Calisto MT                      |       |       | x     |         |       |       |
| Century<br>Gothic               |       |       | x     |         |       |       |
| Century<br>Gothic Bold          |       |       | x     |         |       |       |

|                               |   |   |   |   |   |   |
|-------------------------------|---|---|---|---|---|---|
| Century Gothic Bold Italic    |   |   | X |   |   |   |
| Century Gothic Italic         |   |   | X |   |   |   |
| Comic Sans MS                 |   |   | X | X | X |   |
| Comic Sans MS Bold            |   |   | X | X | X | X |
| Copperplate Gothic Bold       |   |   | X |   |   |   |
| Copperplate Gothic Light      |   |   | X |   |   |   |
| Courier                       | X | X | X | X | X | X |
| Courier New                   | X | X | X | X | X | X |
| Courier New Bold              | X | X | X | X | X | X |
| Courier New Bold Italic       | X | X | X | X | X | X |
| Courier New Italic            | X | X | X | X | X | X |
| Estrangelo Edessa             |   |   |   |   |   | X |
| Franklin Gothic Medium        |   |   |   |   |   | X |
| Franklin Gothic Medium Italic |   |   |   |   | X |   |
| Gautami                       |   |   |   |   |   | X |

|                           |   |   |   |   |   |   |
|---------------------------|---|---|---|---|---|---|
| Georgia                   |   |   |   | X |   | X |
| Georgia Bold              |   |   |   | X |   | X |
| Georgia Bold Italic       |   |   |   | X |   | X |
| Georgia Italic            |   |   |   | X |   | X |
| Georgia Italic Impact     |   |   |   |   |   | X |
| Impact                    |   |   | X | X | X |   |
| Latha                     |   |   |   |   |   | X |
| Lucida Console            |   | X | X | X | X | X |
| Lucida Handwriting Italic |   |   | X |   |   |   |
| Lucida Sans Italic        |   |   | X |   |   |   |
| Lucida Sans Unicode       |   |   | X | X |   | X |
| Marlett                   |   |   | X |   | X |   |
| Matisse ITC               |   |   | X |   |   |   |
| Modern                    | X | X | X | X |   |   |
| Modern MS Sans Serif      |   |   |   |   |   | X |
| MS Sans Serif             | X | X | X | X | X | X |
| MS Serif                  | X | X | X | X | X |   |
| Mv Boli                   |   |   |   |   |   | X |
| News Gothic MT            |   |   | X |   |   |   |
|                           |   |   |   |   |   |   |

|                                     |   |   |   |   |   |   |
|-------------------------------------|---|---|---|---|---|---|
| News Gothic<br>MT Bold              |   |   | X |   |   |   |
| News Gothic<br>MT Italic            |   |   | X |   |   |   |
| OCR A<br>Extended                   |   |   | X |   |   |   |
| Palatino<br>Linotype                |   |   |   | X |   | X |
| Palatino<br>Linotype Bold           |   |   |   | X |   | X |
| Palatino<br>Linotype Bold<br>Italic |   |   | X |   | X |   |
| Palatino<br>Linotype Italic         |   |   |   | X |   | X |
| Roman                               |   | X |   | X |   | X |
| Script                              |   | X |   | X |   | X |
| Small Fonts                         |   | X |   | X |   | X |
| Smallfonts                          | X |   | X |   | X |   |
| Symbol                              | X | X | X | X | X | X |
| Tahoma                              |   |   | X | X | X | X |
| Tahoma Bold                         |   |   | X | X | X | X |
| Tempus Sans<br>ITC                  |   |   | X | X |   |   |
| Times New<br>Roman                  | X | X | X | X | X | X |
| Times New<br>Roman Bold             | X | X | X | X | X | X |
|                                     |   |   |   |   |   |   |

|                                 |   |   |   |   |   |   |
|---------------------------------|---|---|---|---|---|---|
| Times New Roman Bold Italic     | X | X | X | X | X | X |
| Times New Roman Italic          | X | X | X | X | X | X |
| Trebuchet                       |   |   |   |   | X |   |
| Trebuchet Bold                  |   |   |   |   | X |   |
| Trebuchet Bold Italic           |   |   |   |   | X |   |
| Trebuchet Italic                |   |   |   |   | X |   |
| Trebuchet MS                    |   |   |   | X |   | X |
| Trebuchet MS Bold               |   |   |   | X |   | X |
| Trebuchet MS Bold Italic        |   |   |   | X |   | X |
| Trebuchet MS Italic             |   |   |   | X |   | X |
| Tunga                           |   |   |   |   |   | X |
| Verdana (included with MSIE 3+) |   |   | X | X | X | X |
| Verdana Bold                    |   |   | X | X | X | X |
| Verdana Bold Italic             |   |   | X | X | X | X |
| Verdana Italic                  |   |   | X | X | X | X |
| Webdings                        |   |   | X | X | X | X |
| Westminster                     |   |   | X |   | X | X |

|           |   |   |  |   |  |   |
|-----------|---|---|--|---|--|---|
| Wingdings | x | x |  | x |  | x |
| WST_Czech |   |   |  |   |  | x |
| WST_Engl  |   |   |  |   |  | x |
| WST_Fren  |   |   |  |   |  | x |
| WST_Germ  |   |   |  |   |  | x |
| WST_Ital  |   |   |  |   |  | x |
| WST_Span  |   |   |  |   |  | x |
| WST_Swed  |   |   |  |   |  | x |

*List compiled by [KaysKreations](#) and [AutoIt Team](#).*

# Send Messages to a Window or Its Controls

## by Rajat

This page discusses the [PostMessage](#) and [SendMessage](#) commands and will answer some questions like:

"How do I press a button on a minimized window?"

"How do I select a menu item when [MenuSelect](#) doesn't work with it?!"

"This is a skinnable window.... how do I send a command that works every time?"

"and what about **hidden** windows?!"

Requirements: Winspector Spy (<http://www.softpedia.com/get/Security/Security-Related/Winspector.shtml>)

As a first example, note that [MenuSelect](#) fails to work with the menu bar on Outlook Express's "New Message" window. In other words, this code will not work:

```
MenuSelect, New Message,, &Insert, &Picture...
```

But [PostMessage](#) can get the job done:

```
PostMessage, 0x111, 40239, 0, , New Message
```

Works like a charm! But what the heck is that? 0x111 is the hex code of [wm\\_command message](#) and 40239 is the code that this particular window understands as menu-item 'Insert Picture' selection. Now let me tell you how to find a value such as 40239:

1. Open Winspector Spy and a "New Message" window.
2. Drag the crosshair from Winspector Spy's window to "New Message" window's titlebar (the portion not covered by Winspector Spy's overlay).
3. Right click the selected window in the list on left and select 'Messages'.
4. Right click the blank window and select 'Edit message filter'.
5. Press the 'filter all' button and then dbl click 'wm\_command' on the list on left. This way you will only monitor this message.
6. Now go to the "New Message" window and select from its menu bar: Insert > Picture.
7. Come back to Winspector Spy and press the traffic light button to pause monitoring.
8. Expand the wm\_command messages that've accumulated (forget others if any).
9. What you want to look for (usually) is a code 0 message. sometimes there are wm\_command messages saying 'win activated' or 'win destroyed' and other stuff.. not needed. You'll find that there's a message saying 'Control ID: 40239' ...that's it!
10. Now put that in the above command and you've got it! It's the wParam value.

For the next example I'm taking Paint because possibly everyone will have that. Now let's say it's an app where you have to select a tool from a toolbar using AutoHotkey; say the dropper tool is to be selected.

What will you do? Most probably a mouse click at the toolbar button, right? But toolbars can be moved and hidden! This one can be moved/hidden too. So if the target user has done any of this then your script will fail at that point. But the following command will still work:

```
PostMessage, 0x111, 639,,,untitled - Paint
```

Another advantage to `PostMessage` is that the Window can be in the background; by contrast, sending mouse clicks would require it to be active.

Here are some more examples. Note: I'm using WinXP Pro (SP1) ... if you use a different OS then your params may change (only applicable to apps like Wordpad and Notepad that come with windows; for others the params shouldn't vary):

```
;makes writing color teal in Wordpad
PostMessage, 0x111, 32788, 0, , Document -
WordPad
```

```
;opens about box in Notepad
PostMessage, 0x111, 65, 0, , Untitled - Notepad
```

```
;toggles word-wrap in Notepad
PostMessage, 0x111, 32, 0, , Untitled - Notepad
```

```
;play/pause in Windows Media Player
PostMessage, 0x111, 32808, 0, , Windows Media
Player
```

```
;suspend the hotkeys of a running AHK script
DetectHiddenWindows, on
PostMessage, 0x111, 65305,,, MyScript.ahk -
AutoHotkey ; Use 65306 to Pause vs. Suspend.
```

This above was for PostMessage. [SendMessage](#) works the same way but additionally waits for a return value, which can be used for things such as getting the currently playing track in Winamp (see [Automating Winamp](#) for an example).

Here are some more notes:

- The note above regarding OS being XP and msg values changing with different OSes is purely cautionary. if you've found a msg that works on your system (with a certain version of a software) then you can be sure it'll work on another system too with the same version of the software. In addition, most apps keep these msg values the same even on different versions of themselves (e.g. Windows Media Player and Winamp).
- If you've set the filter to show only wm\_command in Winspector Spy and still you're getting a flood of messages then right click that message and select hide (msg name)... you don't want to have a look at a msg that appears without you interacting with the target software.
- The right pointing arrow in Winspector Spy shows the msg values and the blurred left pointing arrows show the returned value. A 0 (zero) value can

by default safely be taken as 'no error' (use it with SendMessage, the return value will be in %ErrorLevel%).

- For posting to hidden windows add this to script:

```
DetectHiddenWindows, On
```

Note: There are apps with which this technique doesn't work. I've had mixed luck with VB and Delphi apps. This technique is best used with C, C++ apps. With VB apps the 'LParam' of the same command keeps changing from one run to another. With Delphi apps... the GUI of some apps doesn't even use wm\_command. It probably uses mouse pos & clicks.

Go and explore.... and share your experiences in the AutoHotkey Forum.  
Feedback is welcome!

This tutorial is not meant for total newbies (no offense meant) since these commands are considered advanced features. So if after reading the above you've not made heads or tails of it, please forget it.

-Rajat