



©1999-2010 Jonathan Bennett & [El Equipo de AutoIt](#)

*Tutorial transcripto a pdf por DuNeD@i
Créditos especiales a Expermicid
www.udtools.net*

Introducción

Autolt v3, es un lenguaje scripting como-BASIC, [freeware](#), diseñado para automatizar la Interfaz gráfica de Windows y scripting en general. También usa una combinación de teclas simuladas, movimientos del mouse y manipulación de ventanas/controles para automatizar tareas en una manera que no es posible o práctica con otros lenguajes (ej. VBScript y SendKeys). Autolt es también muy pequeño, contiene a sí mismo, sin dependencias y es ejecutable en todas las versiones de Windows sin requerir de los molestos "runtimes"!

Autolt fué inicialmente diseñado para situaciones repetitivas de PC y situaciones para automatizar fiablemente y configurar cientos de PCs. Con el tiempo ha venido a ser un lenguaje poderoso que soporta expresiones complejas, funciones de usuario, bucles y cada cosa que los scripters veteranos podrían esperar.

Características principales:

- Fácil para aprender, semejante a la sintaxis de BASIC
- Simula teclas presionadas y movimientos de mouse
- Manipula ventanas y procesos
- Interactúa con todos los controles estándares de ventanas
- Los scripts pueden ser compilados en ejecutables
- Crear Interfaces Gráficas de Usuario (IGU en inglés, GUIs)
- Soporte de COM
- Expresiones Regulares
- Llamada externas directamente de un DLL y funciones del API de Windows
- Funciones Scriptable RunAs
- Ayuda detallada y una gran comunidad de soporte en los foros
- Compatible con Windows 95 / 98 / ME / NT4 / 2000 / XP / 2003 / Vista / 2008
- Soporte Unicode y soporte x64
- Digitalmente firmado para su tranquilidad
- Trabaja con el Control de Usuario de Windows Vista (UAC)

Autolt ha sido diseñado para ser lo más pequeño posible y sin archivos .dll externos o entradas de registro requeridas haciéndolo más seguro para usar en servidores. Los scripts pueden ser compilados en un ejecutable estándar con la herramienta [Aut2Exe](#).

También es suplido con un combinado de la versión COM y DLL de Autolt llamado AutoltX que permite añadir características únicas de Autolt en sus scripts favoritos o lenguajes de programación!

Y lo mejor de todo, Autolt continúa siendo **GRATUITO** - pero si usted desea apoyar el tiempo, dinero y esfuerzo invertidos en el proyecto y el hospedaje web entonces usted puede donar [desde la página en Autolt](#).

Características en Detalle

Sintaxis como-Basic y conjunto de funciones avanzadas

Autolt tiene una sintaxis similar a BASIC, lo que significa que personas que nunca han escrito un script o utilizado un lenguaje de alto nivel deberían ser capaz de comprenderlo fácilmente.

A pesar de que comenzó la vida como una simple herramienta de automatización, Autolt ahora cuenta con funciones y características que le permiten ser utilizado como un lenguaje de scripting de propósito general (con impresionante automatización por supuesto!). Entre las características del lenguaje incluyen:

- El alto-nivel usual de elementos para funciones, bucles y interpretación de expresiones
- Una cantidad asombrosa de funciones para manejo de cadenas y funciones de expresiones regulares compatibles con Perl (mediante la biblioteca [PCRE](#)).
- Soporte COM
- Llamada Win32 y DLL APIs de terceros
- Una extensa librería de funciones, en continuo crecimiento, aportada por la comunidad de contribuyentes

Editor integrado con Sintaxis resaltada

Autolt viene con un versión personalizada "lite" de SciTE que hace fácil la edición de escripts. Los usuarios también pueden descargar una versión completa de SciTE que incluye herramientas adicionales para hacer las cosas aún más fácil.

Pequeño e Independiente

Autolt es una aplicación muy pequeña e independiente, no depende de runtimes masivos como .NET o VB. Todo lo que usted necesita para ejecutar secuencias de comandos es el intérprete de Autolt (Autolt3.exe) y el script claro. Los scripts también pueden ser codificados en ejecutables con el compilador incorporado de scripts: [Aut2Exe](#).

Soporte Internacional y 64-bit

Autolt está diseñado completamente para soportar Unicode y también incluye versiones x64 de todos los componentes principales! Qué tantos otros lenguajes de scripts que son gratuitos pueden decir sobre eso?

Simulación de teclas y mouse

Se ha dedicado mucho tiempo optimizando las funciones de simulación de teclas y mouse para que sean lo más precisas posibles en todas las versiones de Windows. Todas las rutinas de mouse y teclado son altamente configurables, tanto en términos de simulación de "velocidad" y funcionalidad.

Manejo de Ventanas

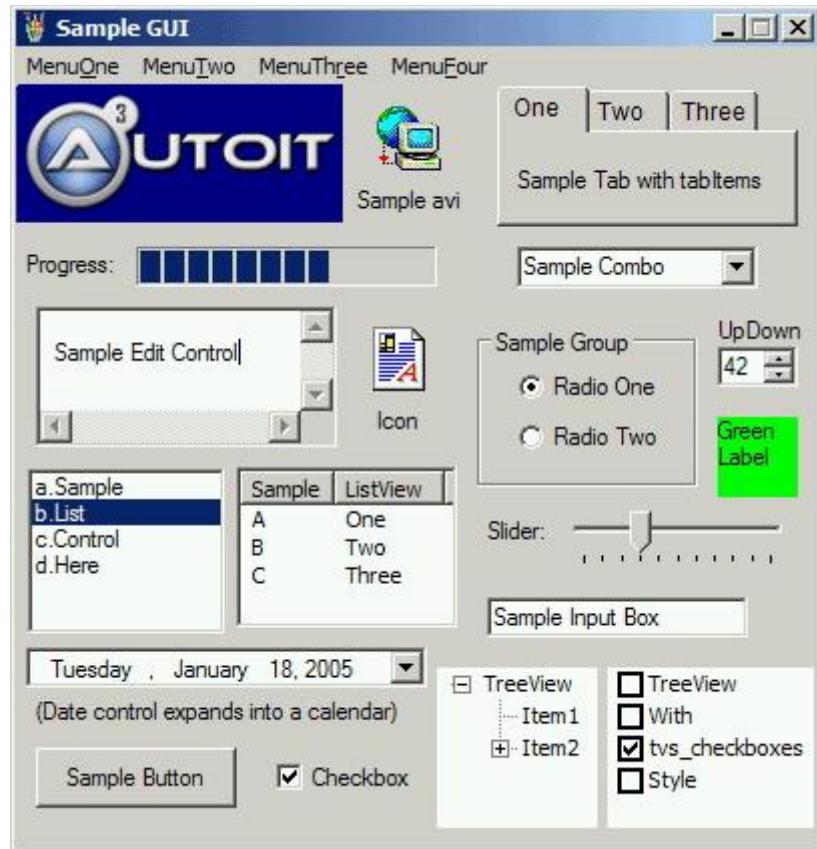
Uste puede mover, ocultar, mostrar, reajustar, activar, cerrar y mucho más con ventanas. Las ventanas pueden ser referenciadas por título, texto en la ventana, tamaño, posición, clases y manejadores internos del API de Win32.

Controles

Directamente obtener información e interactuar con cajas de edición, checkboxes, cajas de listas, combos, botones, barras de estado sin el riesgo de perderse a través de envío de teclas. Aún trabajando con controles en ventanas que no están activas!

Interfaces Gráficas de Usuario (GUIs)

AutoIt v3 también le permitirá crear algunas GUIs complejas - como las que se ven abajo!



Y mucho, mucho mas!...

Licencia de Software

Author : Jonathan Bennett y el equipo de AutoIt

WWW : <http://www.autoitscript.com/autoit3/>

Email : support at autoitscript dot com

ACUERDO DE LICENCIA DE USUARIO DE ESTE SOFTWARE (CLUF)

Esta licencia de usuario final ("CLUF") es un acuerdo legal entre usted (ya sea un individuo o una entidad única) y el mencionado autor de este software para el producto de software identificado más arriba, que incluye software y puede incluir medios asociados, materiales impresos, y "en línea" o documentación electrónica ("PRODUCTO DE SOFTWARE"). Al instalar, copiar, o de otro modo usar el PRODUCTO DE SOFTWARE, usted acepta que quedará vinculado por los términos de este EULA. Si usted no está de acuerdo con los términos de este EULA, no instale o use el PRODUCTO DE SOFTWARE.

LICENCIA DEL PRODUCTO DE SOFTWARE

El PRODUCTO DE SOFTWARE está protegido por leyes de copyright y tratados internacionales de derechos de autor, así como otras leyes de propiedad intelectual y los tratados. El PRODUCTO DE SOFTWARE es licenciado, no vendido.

La definición de PRODUCTO DE SOFTWARE no incluye incluye todos los archivos generados por el PRODUCTO DE SOFTWARE, tales como archivos de script compilado en forma de ejecutables.

1. CONCESIÓN DE LICENCIA.

Este CLUF le otorga los siguientes derechos:

De Instalación y Uso. Puede instalar y utilizar un número ilimitado de copias del PRODUCTO DE SOFTWARE.

Reproducción y Distribución. Usted podrá reproducir y distribuir un número ilimitado de copias del PRODUCTO DE SOFTWARE en su totalidad o en parte, cada copia deberá incluir todos los derechos de autor y marcas comerciales, y deberá ir acompañada de una copia de este EULA. Copias del PRODUCTO DE SOFTWARE se puede distribuir como un producto independiente o incluido con su propio producto.

Uso Comercial. Usted puede vender con fines de lucro y libremente distribuir los scripts y / o scripts compilado que se crearon con el PRODUCTO DE SOFTWARE.

Ingeniería inversa. Usted no puede aplicar ingeniería inversa o desensamblar el PRODUCTO DE SOFTWARE o compilar scripts que fueron creados con el PRODUCTO DE SOFTWARE.

2. DERECHOS DE AUTOR.

Todos los títulos y derechos de autor en y para el PRODUCTO DE SOFTWARE (incluyendo pero no limitado a las imágenes, fotografías, animaciones, vídeo, audio, música, texto y "applets" incorporados en el PRODUCTO DE SOFTWARE), el material impreso que acompaña, y las copias del PRODUCTO DE SOFTWARE son propiedad del autor de este Software. El PRODUCTO DE SOFTWARE está protegido por las leyes y disposiciones de tratados internacionales. Por lo tanto, usted debe tratar el PRODUCTO DE SOFTWARE como cualquier otro material con derechos de autor.

DIVERSOS

Si adquirió este producto en el Reino Unido, este CLUF se regirá por las leyes del Reino Unido.

Si este producto fue adquirido fuera del Reino Unido podrá aplicar la ley local.

Si usted tiene alguna pregunta acerca de este CLUF, o si desea ponerse en contacto con el autor de este software por cualquier razón, póngase en contacto con él / ella en la dirección de correo electrónico mencionado en la parte superior de este EULA.

GARANTÍA LIMITADA

SIN GARANTÍAS.

El Autor de este Software renuncia expresamente a cualquier garantía para el PRODUCTO DE SOFTWARE. El PRODUCTO DE SOFTWARE y toda la documentación relacionada se proporciona "tal cual" sin garantía de ningún tipo, ya sea expresa o implícita, incluyendo, sin limitación, las garantías implícitas de comerciabilidad, adecuación para un propósito particular, o no infracción. La totalidad del riesgo resultante del uso o rendimiento del PRODUCTO DE SOFTWARE sigue con usted.

NO HAY RESPONSABILIDAD POR DAÑOS.

En ningún caso el autor de este software podrá ser considerada responsable de ningún daño (incluyendo, sin limitación, daños por pérdida de beneficios comerciales, interrupción del negocio, la pérdida de información comercial, o cualquier otra pérdida pecuniaria) que se derive del uso o incapacidad para utilizar este producto, incluso si el autor de este software ha sido advertido de la posibilidad de tales daños. Debido a que algunos estados o jurisdicciones no permiten la exclusión o limitación de responsabilidad por daños consecuentes o incidentales, la limitación anterior podría no aplicarse a usted.

[FINAL DE LA LICENCIA

Estructura del Directorio de Instalación

El instalador de Autolt crea una estructura de directorios (habitualmente situada en `\Archivos de Programa\Autolt3`) recopilada en la siguiente tabla. El instalador también crea Accesos Directos en el Menú Inicio, pero otros archivos no son agregados ni modificados.

Archivos y Directorios	Descripción
(Archivos en el Nivel-Superior)	
Autolt3.exe	El programa principal de Autolt y único archivo requerido como intérprete de los scripts!
Autolt3_x64.exe	Versión x64 de Autolt (si es instalada).
AU3Info.exe	La Herramienta Autolt para Información de Ventanas .
AU3Info_x64.exe	Versión x64 de Au3Info (si es instalada).
AU3InfoA.exe	La versión ANSI de Au3Info.exe para Windows 9x.
AU3Check.exe	El analizador de sintaxis de Autolt.
Autolt.chm	El archivo que está leyendo ahora, refiriendo en el mismo al UDFs3.chm
Uninstall.exe	El desinstalador de Autolt. Probablemente nunca necesite usarlo.
Autolt v3 Website.url	Un acceso directo al sitio http://www.autoitscript.com/autoit3/
Aut2Exe	
Icons\	Contiene íconos usados por Explorer para el tipo de archivo .au3.
Aut2Exe.exe	El compilador de script.
Aut2Exe_x64.exe	Versión x64 de Aut2Exe (si es instalada).
AutoltSC.bin	Paquete ejecutable para scripts compilados

	AutoItSC_x64.bin	Paquete ejecutable para scripts compilados bajo x64.
	UPX.exe	El compresor UPX (disminuye el tamaño de los archivos ejecutables).

Examples

	GUI\	Contiene ejemplos de GUIs escritos en AutoIt.
	Helpfile\	Contiene muchos ejemplos de Scripts usados en el Archivo de Ayuda.

Extras

	AutoUpdateIt\	Contiene un script que le facilita obtener la última versión de AutoIt.
	Editors\	Contiene definiciones para sintaxis coloreada de algunos editores de texto populares.
	Exe2Aut\	Contiene utilidades para convertir scripts compilados a su código fuente.
	SQLite\	Contiene ejecutables de la línea de comandos SQLite y su respectivo archivo de ayuda.
	v2_to_v3_Converter\	Contiene una herramienta para conversión de scripts desde la v2.64 a la sintaxis de AutoIt v3.

Icons

	Contiene íconos usados por Explorer para el tipo de archivo .au3
--	--

Include

	Contiene los archivos estándar de inclusión (funciones de usuario pre-escritas). Vea la librería de funciones
--	---

AutoItX

	Contiene una versión DLL de AutoIt v3 que proporciona un subconjunto de
--	---

	características de AutoIt a través de un ActiveX/COM y una interfaz DLL.
SciTe	
	Contiene una versión liviana de SciTe que permite resaltar sintaxis con color.

Una vez más le recordamos que para poder correr scripts de AutoIt, el único archivo requerido es **AutoIt3.exe**. Si usted compila un script, el ejecutable generado no requiere que el usuario tenga AutoIt instalado en su máquina para poder ejecutarlo.

(excepción: Bajo Windows NT 4 es necesario que el archivo PSAPI.dll esté en la ruta o directorio de AutoIt para que trabajen las funciones relativas a Process...())

Claves del Registro

El instalador de AutoIt crea claves de Registro en *HKEY_LOCAL_MACHINE\Software\AutoIt v3* y *HKEY_CURRENT_USER\Software\AutoIt v3*. Las claves no son usadas/creadas cuando las utilidades de AutoIt están ejecutándose sobre máquinas que no tienen una instalación completa--AutoIt está "limpio" para correr sobre servidores, etc.

La tabla aquí debajo le muestra las claves del registro predeterminadas (o típicas). Las claves en cursiva no son creadas por el instalador en si mismo, sino por la primera ejecución de la utilidad correspondiente:

HKEY_LOCAL_MACHINE\SOFTWARE\AutoIt v3\			
AutoIt			
	(Predeterminado)	REG_SZ	(valor no establecido)
	InstallDir	REG_SZ	C:\Archivos de Programa\AutoIt3
	Version	REG_SZ	Número de la Versión
HKEY_CURRENT_USER\Software\AutoIt v3\			
Aut2Exe			
	(Predeterminado)	REG_SZ	(valor no establecido)
	AllowDecompile	REG_DWORD	0x1
	LastCompression	REG_DWORD	0x2
	LastExeDir	REG_SZ	Mis documentos

	LastIcon	REG_SZ	
	LastIconDir	REG_SZ	C:\Archivos de Programa\Autolt3\Aut2Exe\Icons
	LastScriptDir	REG_SZ	Mis documentos
AutoUpdateIt			
	(Predeterminado)	REG_SZ	(valor no establecido)
	DoneOption	REG_SZ	Notificar
	DownloadDir	REG_SZ	C:\Descargas\PorEjemplo\
Exe2Aut			
	(Predeterminado)	REG_SZ	(valor no establecido)
	LastExeDir	REG_SZ	C:\PorEjemplo\
	LastScriptDir	REG_SZ	
AU3Info			
	Predeterminado	REG_SZ	(valor no establecido)
	AlwaysOnTop	REG_DWORD	0x1
	ColorMode	REG_DWORD	0x1
	CoordMode	REG_DWORD	0x1
	HighlightColor	REG_DWORD	0x0
	HighlightControls	REG_DWORD	0x1
	Magnify	REG_DWORD	0x0
	WinH	REG_DWORD	0x01c2
	WinW	REG_DWORD	0x012c

	WinX	REG_DWORD	0x0064
	WinY	REG_DWORD	0x0064

Preguntas Más Frecuentes (FAQ)

Esta sección da información de las preguntas más frecuentes del [foro](#). Si no encuentras la respuesta aquí, entonces sería bueno revisar en el [foro](#).

Preguntas

1. [¿Porque mi antiguo script de AutoIt v2.64 no funciona en v3?](#)
2. [¿No es más difícil v3 que las versiones anteriores?](#)
3. [¿Cómo puedo convertir mis scripts de v2.64 a v3?](#)
4. [¿Dónde está el comando "goto"?](#)
5. [¿Cómo puedo ejecutar un programa de DOS dentro en AutoIt?](#)
6. [¿Porqué sólo puedo usar Run\(\) para ejecutar archivos .exe y .com? ¿Qué sobre .msi / .txt y otros?](#)
7. [¿Porqué tengo errores cuando intento usar comillas dobles \(""\) ?](#)
8. [¿Qué significan los parámetros de ventana "título" y "texto"?](#)
9. [¿Porqué no puedo mostrar una variable usando "Mi var es \\$variable"?](#)
10. [Cuando yo uso Send\(\) para enviar una variable ¿que posibilidades pueden ocurrir?](#)
11. [¿Qué diferencia hay entre el valor de retorno y @error?](#)
12. [¿Cómo puedo puedo terminar mi script con un hot-key?](#)
13. [¿Cómo puedo usar un ícono personal cuando compilo mis scripts?](#)
14. [¿Cómo puedo asegurarme que sólo una copia de mi script se esté ejecutando?](#)
15. [¿Cuáles son las actuales limitaciones técnicas de AutoIt v3?](#)
16. [Yo obtengo un símbolo de imagen perdida en los ejemplos del archivo de Ayuda.](#)

1. ¿Porque mi antiguo script de AutoIt v2.64 no funciona en v3?

v3 tiene una estructura de lenguaje diferente a v2.64.

Versiones anteriores de AutoIt estaban bien para lo que fueron diseñadas - escribiendo scripts sencillos para ayudar con las instalaciones de software. Tiempo después la gente empezó a usarlo para cosas generales y tareas más complicadas. La vieja sintaxis y estructura hizo posible esto pero muy muy difícil de mantenerlo. La decisión fué hacer un AutoIt más satisfactorio para automatización de tareas en general y lograr más que un estándar como el lenguaje Basic fué hecho. Esto también significa que si tú ya conoces un lenguaje de scripts te adaptarás fácilmente con AutoIt v3.

[Regresar al Principio](#)

2. ¿No es más difícil v3 que las versiones anteriores?

No. De hecho en muchas instancias es más fácil que versiones anteriores como tú no tienes que forzar el lenguaje a hacer cosas que nunca fueron diseñadas para que haga. También usa

un lenguaje familiar al lenguaje BASIC, y BASIC es bien conocido para principiantes...bueno...básico :)

La vasta mayoría de viejas versiones de scripts de Autolt fueron enfocadas a instalaciones de software y clickeos como "Siguiente" en muchos cuadros de diálogo. La mayoría de estos scripts pueden ser convertidos a v3 simplemente añadiendo algunos paréntesis aquí y allá. Aquí hay un ejemplo de un script v2 y v3 (simulando una instalación de software con algunas cajas de diálogos que tiene los botones Siguiente y Finalizar)

```
; v2.64 Script
WinWaitActive, Welcome, Welcome to the XSoft installation
Send, !
WinWaitActive, Choose Destination, Please choose the
Send, !
WinWaitActive, Ready to install, Click Next to install
Send, !
WinWaitActive, Installation Complete, Click Finish to exit
Send, !
WinWaitClose, Installation Complete

; v3 Script
WinWaitActive("Welcome", "Welcome to the XSoft installation")
Send("!")
WinWaitActive("Choose Destination", "Please choose the")
Send("!")
WinWaitActive("Ready to install", "Click Next to install")
Send("!")
WinWaitActive("Installation Complete", "Click Finish to exit")
Send("!")
WinWaitClose("Installation Complete")
```

Ahora, eso no estuvo mal! :) Como todas las "cadenas de texto" están encerradas en comillas no hay más problemas con espacios antes o después en el texto. Aquí también hay un soporte para muchos [editores de texto](#) así que al escribir scripts v3 tú puedes tener la sintaxis resaltada lo cual lo hace más fácil.

[Regresar al Principio](#)

3. ¿Cómo puedo convertir mis scripts de v2.64 a v3?

La primera cosa que uno debe preguntarse es "Necesito convertir mi script?". v2.64 continuará siendo descargable y soportado así que no sería necesario actualizar todos los scripts solo por ese temor - bueno a menos que quieras :)

Aquí hay una sección en el archivos de ayuda que muestra como los comandos de v2 y v3 están relacionados - haz un click [aquí](#) para ver la página.

Uno de los autores de la versión 3 escribió una utilidad para convertir scripts v2 a v3. La conversion es muy buena, a menos que tu código sea un nido de ratas de "goto" :) Tú puedes encontrar el convertidor en el directorio de "Extras" (Start \ AutoIt v3 \ Extras - o revisa donde has instalado AutoIt v3).

[Regresar al Principio](#)

4. ¿Dónde está el comando "goto"?

Se fué. Es malo. No, no preguntes porqué - sólo es así. Es como ese montón de piedras que encontraron en el microondas al final de la película [Time Bandits](#) :)

Muchas de las características de AutoIt v3 tiene los "loops" comunes de uso actual y esos Goto no son requeridos. Revisa [While](#), [Do](#), [For](#), [ExitLoop](#), [ContinueLoop](#) y [Funciones](#) para ver la manera moderna de hacer las cosas :) También puedes ver en el archivo de ayuda estas secciones [bucles](#), [sentencias condicionales](#) y [funciones](#). Te prometo, una vez que tienes a mano tales cosas, virtualmente serás capaz de escribir en otros lenguajes en poco tiempo.

Para iniciar, el más básico uso de Goto en la versión 2.64 como un infinito bucle como este:

```
:mylabel  
...do something...  
...and something else...  
goto, mylabel
```

Una sencillo ejemplo en v3 usando el bucle While que siempre es "VERDADERO".

```
While 1 = 1  
...do something...  
...do something else...  
Wend
```

[Regresar al Principio](#)

5. [¿Cómo puedo ejecutar un programa de DOS dentro en AutoIt?](#)

Si quieras ejecutar comandos de DOS, como el comando "Dir", entonces debes ejecutarlo a través del intérprete (command.com o cmd.exe dependiendo de tu SO). El macro [@Comspec](#) contiene la ubicación correcta de este archivo. Deberías usar la función [RunWait\(\)](#) como para la típica espera de muchos programas en DOS antes de que aparezca la siguiente línea del script. Aquí hay un ejemplo ejecutando el comando de DOS, "Dir" en la unidad C: (efectivamente ejecutándose el comando **command.com /c Dir C:**)

```
RunWait(@COMSPEC & " /c Dir C:\")
```

[Regresar al Principio](#)

6. ¿Porqué sólo puedo usar Run() para ejecutar archivos .exe y .com? ¿Qué sobre .msi / .txt y otros?

Sólo algunas extensiones son usualmente "ejecutables" - éstas son .exe, .bat, .com, .pif. Otras extensiones como .txt y .msi son actualmente ejecutadas con otro programa. Cuando se hace un doble click en un archivo como "myfile.msi" lo que pasa es que en el fondo es ejecutado "msiexec.exe myfile.msi". Así para que ejecutar un archivo .msi de AutoIt hay que hacer lo siguiente:

```
RunWait("msiexec myfile.msi")
```

O ejecutar el comando "start" el cual automáticamente sabe como deberá ejecutarse el archivo:

```
RunWait(@COMSPEC & "/c Start myfile.msi")
```

O usa la función interna de AutoIt llamada ShellExecuteWait, que automáticamente te ahorrará más tiempo:

```
ShellExecuteWait("myfile.msi")
```

[Regresar al Principio](#)

7. ¿Porqué tengo errores cuando intento usar comillas dobles ("") ?

Si quieres usar doble comillas dentro de una cadena de texto o string entonces debes "doblar o duplicar comillas". Así que para cada una de las comillas que quieras usar, usa las dos. Para entender mejor, ver el siguiente ejemplo:

Una palabra en "esta" oración tiene comillas! Se debe hacer esto:

```
$var = "Una palabra en ""esta"" oración tiene comillas!"
```

o usar comillas simples:

```
$var = 'Una palabra en "esta" oración tiene comillas!'
```

[Regresar al Principio](#)

8. ¿Qué significan los parámetros de ventana "título" y "texto"?

Hay una descripción detallada de esto [aquí](#).

[Regresar al Principio](#)

9. ¿Porqué no puedo mostrar una variable usando "Mi var es \$variable"?

Si tienes una variable llamada \$msg y quieres imprimir dentro de un MsgBox como la siguiente línea, no funcionará:

```
MsgBox(0, "Ejemplo", "Mi variable es $msg")f
```

Se mostrará **Mi variable es \$msg**. Lo que necesitas es decirle a AutoIt: une la cadena de texto y la variable juntos, esto se logra usando el [operador &](#):

```
MsgBox(0, "Ejemplo", "Mi variable es " & $msg)
```

Avanzado: Si tienes muchas variables para añadir a la cadena de texto entonces hay una función mas útil para este caso llamada [StringFormat\(\)](#). Por ejemplo, si se desea insertar de \$var1 a \$var5 en una cadena entonces sería más fácil esto:

```
$msg = StringFormat("Var1 es %s, Var2 es %s, Var3 es %s, Var4 es %s, Var5 es %s", $var1,  
$var2, $var3, $var4, $var5)  
MsgBox(0, "Ejemplo", $msg)
```

[Regresar al Principio](#)

10. [Cuando yo uso Send\(\) para enviar una variable ¿que posibilidades pueden ocurrir?](#)

Si se está [enviando](#) el contenido de una variable entonces será útil que si contiene caracteres especiales como ! ^ + {SPACE} entonces estos serán convertidos a keystrokes especiales - que raramente se necesita. Para superar esto usa el modo RAW de [Send\(\)](#) que no traduce las teclas especiales:

```
Send($myvar, 1)
```

[Regresar al Principio](#)

11. [¿Qué diferencia hay entre el valor de retorno y @error?](#)

Generalmente un valor devuelto es usado para indicar el éxito de una función. Pero, si una función ya está devolviendo algo (como [WinGetText\(\)](#)) entonces necesitamos tener una manera de trabajar si la función tuvo éxito, así que definimos @error en su lugar.

[Regresar al Principio](#)

12. [¿Cómo puedo puedo terminar mi script con un hot-key?](#)

Ah, muy fácil. Si quieras hacer un script que al presionar cierta tecla termine el script usa la función [HotKeySet\(\)](#) para definir en qué función hará que termine. Esta función de usuario sólo debería contener la palabra clave [Exit](#).

Aquí un ejemplo que causará que al presionar CTRL+ALT+x termine:

```
HotKeySet("^!x", "MyExit")
...
...
; Resto del Script
...
...
Func MyExit()
    Exit
EndFunc
```

[Regresar al Principio](#)

13. ¿Cómo puedo usar un ícono personal cuando compilo mis scripts?

Para eso es necesario ejecutar el programa compilador (Alt + F7 en SciTe) o puedes simplemente dar click-derecho sobre un fichero de script y seleccionar compilar). [Esta página](#) describe el compilador en detalle.

[Regresar al Principio](#)

14. ¿Cómo puedo asegurarme que sólo una copia de mi script se esté ejecutando?

Usa la función _Singleton(). Ver la documentación de las Funciones Definidas de Usuario (UDF) para más información en _Singleton() y cómo usarlo.

[Regresar al Principio](#)

15. ¿Cuáles son las actuales limitaciones técnicas de AutoIt v3?

[Aquí](#) se muestra detalles de los límites técnicos actuales de AutoIt. Por favor nota que algunos límites son teóricamente y tal vez dependa del desempeño o memoria y problemas relacionados antes de llegar a tales límites.

Máxima longitud de una línea simple de un script: 4,095

Máxima longitud de una cadena de texto: 2,147,483,647 caracteres

Rango de número (punto flotante): 1.7E-308 to 1.7E+308 con precisión de 15-dígitos

Rango de número (enteros): 64-bit entero signed

Números Hexadecimales: 32-bit entero signed (0x80000000 a 0x7FFFFFFF)

Arreglos (Arrays): Un máximo de 64 dimensiones y/o un total de 16 millones de elementos

Rango máximo de llamadas en funciones recursivas: 5100 niveles

Máximo número de variables en uso a la vez: No hay límite

Máximo número de funciones definidas de usuario: No hay límite

Máximo número de ventanas GUI: No hay límite

Máximo número de controles GUI: 65532

[Regresar al Principio](#)

16. Yo obtengo un símbolo de imagen perdida en los ejemplos del archivo de Ayuda.

Esto debería ser el botón de abrir que activa los ejemplos en el archivo de Ayuda. Esta cuestión es que el control "hhctrl.ocx" no está apropiadamente registrado o está corrupto.

Intenta registrarlo desde el símbolo de sistema o cmd ejecutando "regsvr32 hhctrl.ocx" o revisando si el archivo continúa siendo válido en el sistema.

Ejecutando Scripts

Cuando comience, Autolt le preguntará para abrir un **archivo script**. Un script es un archivo de texto simple, que contiene palabras claves de Autolt y funciones con las cuales usted crea el código y Autolt lo interpreta. Los archivos scripts pueden ser creados en un editor de texto común y corriente como **notepad.exe**, aunque hay mejores [alternativas](#).

Aunque los scripts de Autolt v3 son simplemente archivos de texto plano, por lo común ellos tienen la extensión **.au3** para lograr diferenciar fácilmente un script de un archivo de texto. Si usted utilizó la instalación completa de Autolt puede ejecutar un script de Autolt haciéndole doble click al archivo. Hay también variadas formas para abrir, editar, o compilar un archivo script si usted da un click-derecho sobre el archivo **.au3**

Aquí hay un ejemplo de script. Le notificamos que el ; se emplea para comentarios (muy similar al **REM** en archivos de lotes del DOS):

```
; Este es mi primer script
MsgBox(0, "¡Mi Primer Script!", "¡Hola Mundo!")
```

Los Scripts más complejos pueden usar **funciones**, que son ubicadas generalmente al final del archivo. Aquí hay un script que utiliza funciones:

```
; Este es mi segundo script (con funciones)
MsgBox(0, "¡Mi segundo script!", "¡Hola desde el script principal!")
PruebaFunc()

Func PruebaFunc()
    MsgBox(0, "¡Mi segundo Script!", "¡Hola desde la función!")
EndFunc
```

Parámetros de la Línea de Comandos

El arreglo especial **\$CmdLine** es inicializado con los parámetros de la linea de comandos ingresados a su script Autolt. Nótese que el nombre del script no es clasificado como un parámetro; en su lugar capture esa información con **@ScriptName**. Un parámetro que contiene espacios debe confinarse entre "comillas". Scripts [compilados](#) aceptan parámetros de la línea de comandos en la misma forma.

\$CmdLine[0] es el número de parámetros
\$CmdLine[1] es el param 1 (después del nombre del Script)
\$CmdLine[2] es el param 2 etc
...
\$CmdLine[\$CmdLine[0]] es una forma de obtener el último parámetro...

nota: los parámetros de línea de comandos son separados por un espacio.

De esta manera, si su script luciera así:

```
Autolt3.exe miscript.au3 param1 "este es otro param"  
$CmdLine[0] equivale a... 2  
$CmdLine[1] equivale a... param1  
$CmdLine[2] equivale a... este es otro param  
@ScriptName equivale a... miscript.au3
```

Cabe agregar que en \$CmdLine hay una variable llamada \$CmdLineRaw que contiene la linea de comandos sin separar, aplicado entonces al ejemplo de arriba:

\$CmdLineRaw *equivale a...* miscript.au3 param1 "este es otro param"

Si el script estuviera compilado debería ser ejecutado como sigue:

```
miscript.exe param1 "este es otro param"  
$CmdLineRaw equivale a... param1 "este es otro param"  
Nótese que $CmdLineRaw únicamente retorna los parámetros.
```

Nota : solamente es posible retornar 63 parámetros con \$CmdLine[...], pero \$CmdLineRaw le permitirá siempre obtener la línea de comandos completa.

Interruptores específicos de Autolt para la Línea de Comandos

Forma1: *Autolt3.exe [/ErrorStdOut] [/Autolt3ExecuteScript] archivo [params ...]*
Ejecuta un Archivo Script de Autolt3

/ErrorStdOut Permite redirigir un error fatal al StdOut, el cual puede ser capturado por una aplicación como el editor Scite. Es posible utilizar este interruptor con un script compilado.

Para ejecutar un Archivo Script estándar de Autolt 'miscript.au3', use el comando:
'Autolt3.exe miscript.au3'

Forma2: *Compilado.exe [/ErrorStdOut] [params ...]*
Ejecuta un archivo Script Autolt3 compilado producido con Aut2Exe.

Forma3: *Compilado.exe [/ErrorStdOut] [/Autolt3ExecuteScript file] [params ...]*
Ejecuta otro archivo script desde un archivo Script compilado Autolt3. Así usted no necesita usar fileinstall con otra copia del Autolt3.exe dentro del mismo archivo compilado.

Forma4: *Autolt3.exe [/ErrorStdOut] /Autolt3ExecuteLine "línea de comando"*
Ejecuta una línea de código.

Para ejecutar una línea simple de código, use el comando:

```
Run(@AutoItExe & '/AutoIt3ExecuteLine "MsgBox(0, "¡Hola Mundo!", "¡Hola!")")'
```

El ícono Tray no será mostrado cuando utiliza /AutoIt3ExecuteLine

NOTA: El uso correcto de comillas- y apóstrofes- es importante, invariablemente **comillas**.

La Forma3 y Forma4 pueden ser deshabilitadas para scripts compilados cuando se utiliza
#NoAutoIt3Execute.

Autolt sobre Windows Vista

Windows Vista proporciona nuevas características de seguridad para restringir la ejecución de archivos que requieran privilegios administrativos. Incluso los usuarios administradores serán advertidos cada vez que un ejecutable realice alguna operación administrativa (como por ejemplo escribir a la llave del registro HK_EY_LOCAL_MACHINE o escribir en el directorio C:\Windows). A esto se lo denomina **Control de la cuenta del Usuario (UAC)**.

De manera predefinida, los scripts de Autolt corren con los permisos de usuarios estándar, pero Autolt fue diseñado para permitirle al autor del código, la opción de "marcar" su script con el fin de decirle a Autolt si para ser ejecutado correctamente, necesitará de privilegios administrativos o no.

Para forzar un script a que intente correr con privilegios administrativos, agregue la directiva **#requireadmin** en el nivel superior de su código, tal como sigue:

```
; Este script requiere privilegios Administrativos completos  
#requireadmin
```

```
MsgBox(0, "Info", "Este script tiene derechos administrativos! ")
```

Cuando el script se ejecuta, Autolt revisará si ya tiene los privilegios administrativos, y en caso de que no fuera así, el sistema operativo enviará una alerta al usuario requiriendo su permiso, tal como se muestra en "Alertas UAC". Si el permiso no es otorgado por el usuario, el script terminará.

Alertas UAC

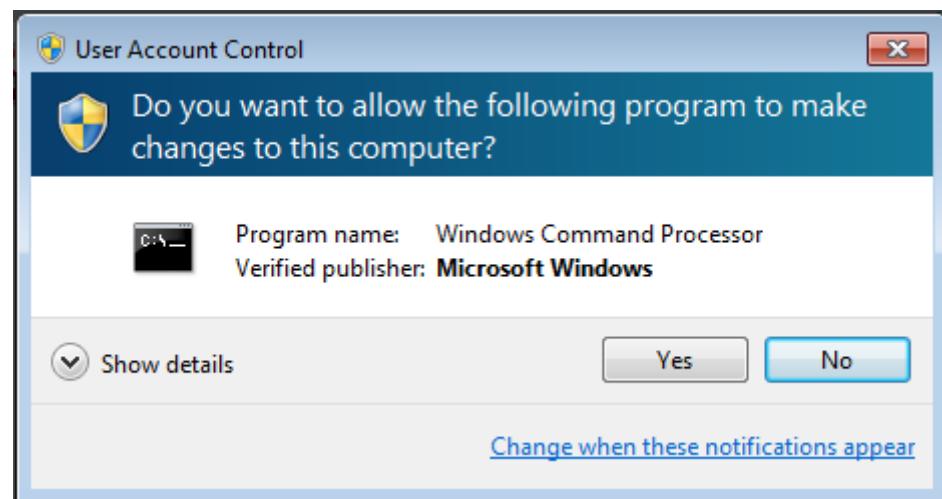
Las alertas que Vista mostrará cuando ejecute un programa con privilegios administrativos se muestran debajo. Los tipos de alertas que se muestren dependerán de si el usuario es un "usuario estándar" o bien un "usuario administrador" (recuerde que incluso los administradores necesitan elevar sus permisos para realizar operaciones administrativas). Nota: Los avisos de alerta mostrados están con la versión digitalmente firmada de Autolt - todas las versiones lanzadas están firmadas, aunque las versiones beta pueden no estarlo, y darán como resultado una advertencia como se muestra debajo en "Scripts Compilados".

Alertas de Usuario Estándar



Un usuario estándar debe seleccionar un nombre de usuario e ingresar una contraseña para poder continuar y correr el script que requiera permisos superiores.

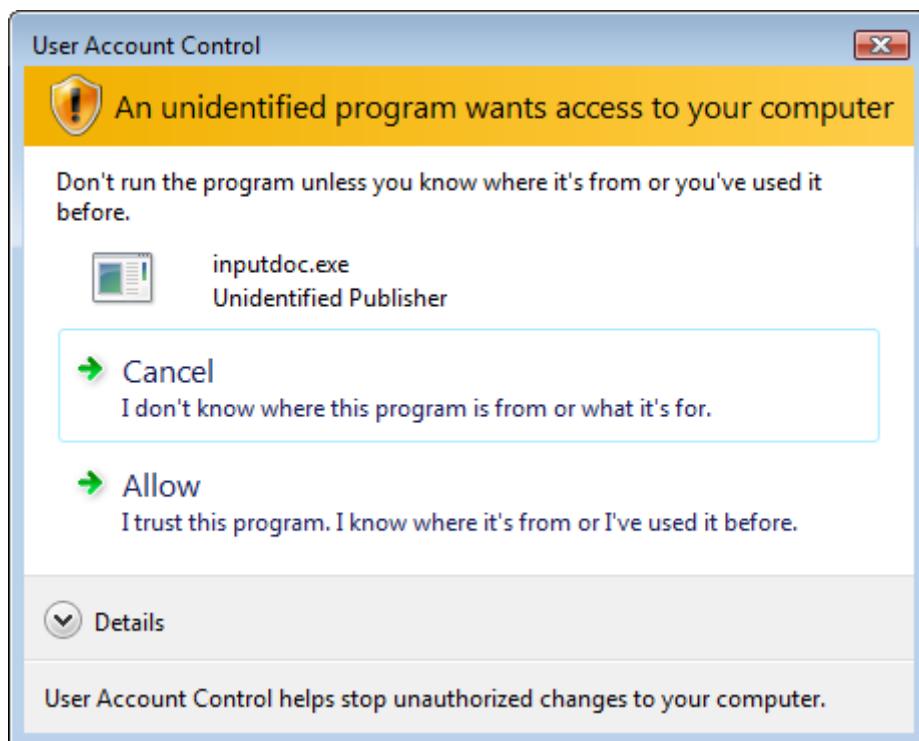
Alertas de Usuario Administrador



Como el usuario ya es administrador del sistema y solamente se requieren permisos superiores, para continuar deberá hacer un clic de confirmación - no es necesario ingresar la contraseña.

Scripts Compilados

Los scripts Compilados (y posiblemente versiones beta de AutoIt) **no** están digitalmente firmadas y por tanto darán un aspecto más serio a la alerta, tal como se muestra a continuación:



El usuario debe efectuar un clic para **Permitir** que continúe (o entrar la contraseña si es un usuario estándar).

Si tiene su propia firma de autentificación para código, podrá firmar sus scripts compilados usted mismo.

Importante: Aunque AutoIt o un script compilado se firme o no ¡debería correr solamente los scripts de fuentes en las cuales usted confíe!

¡Incluso código firmado puede resultar malicioso!

Editores de Script

Los scripts de Autolt son simples archivos de texto que pueden ser editados empleando sólo el Bloc de Notas de Windows. Sin embargo, son muchos los editores gratuitos, o los que disponen de versiones de prueba, que son más efectivos para escribir código, ofrecen sintaxis que resalta las palabras claves y las funciones de Autolt, consiguiendo que el desarrollo sea **mucho** más sencillo a la vez que disminuyen la probabilidad de cometer errores.

El Editor usado actualmente por la mayoría de los usuarios de Autolt es [SciTe](#), El equipo de desarrollo de Autolt creó una versión personalizada de SciTe con sintaxis completa que se resalta mientras escribe, además de integrar varias herramientas de Autolt (como el control de Sintaxis y Ordenamiento de escritura). La versión de SciTe para Autolt puede encontrarla en <http://www.autoitscript.com/autoit3/scite/>

Otros editores recomendados son:

- [TextPad](#)
- [Crimson Editor](#) (gratuito)
- [Source Edit](#) (gratuito)
- [UltraEdit](#)

En los [foros](#) podrá encontrar diversidad de opiniones de los usuarios y los argumentos que cada uno propone sobre cual es el mejor. :)

Autolt proporciona algunos archivos de sintaxis pre-escritos para muchos editores que usted puede encontrar en el directorio **Extra** de la instalación (existe un enlace en el Menú Inicio / Autolt v3 / Extras).

Compilando Scripts con Aut2Exe

Es posible tomar sus scripts *.au3* y compilarlos en un ejecutable **independiente** ; Este ejecutable podrá ser usado sin requerir que AutoIt se encuentre instalado en el sistema ni tener el archivo *AutoIt3.exe* en la máquina. Además, el script a compilar será comprimido y encriptado, contando usted con la opción de añadir archivos adicionales al exe (también comprimidos/criptados), por medio de la función [FileInstall](#) . Incluso, cualquier archivo [#include](#) podrá ser compilado dentro del script sin necesidad de contar con librerías externas.

Precaución: el script a ser compilado debe estar libre de errores en la sintaxis ya que la compilación no lo revisará.

Aut2Exe puede emplearse en tres diferentes maneras:

Método 1 - Menú Inicio

Únicamente disponible si se realizó la instalación completa.

1. Abra el Menú Inicio y diríjase al grupo *AutoIt v3*.
2. Clic en *Script Compiler \ Convert .au3 to .exe*
3. La interfaz principal de Aut2Exe debería aparecer.



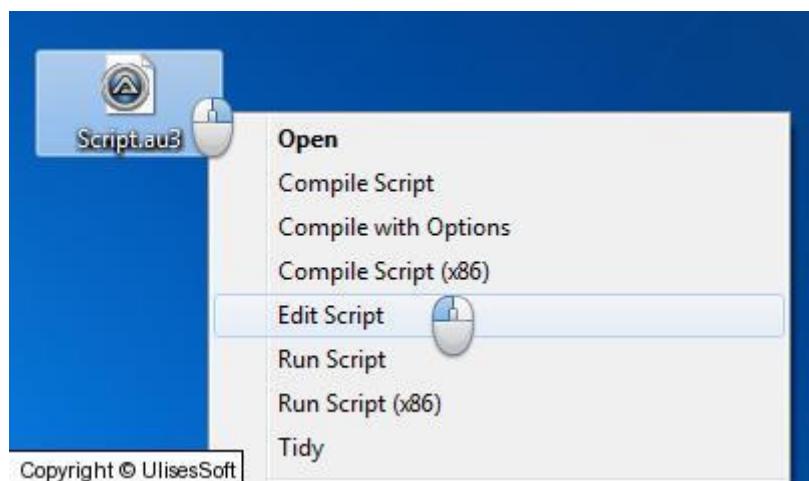
4. Use los botones de *navegar* para seleccionar la ruta de su archivo (.au3) de entrada y el (.exe) de salida.
5. Si usted desea cambiar el ícono del .exe - sencillamente examine hasta hallar el ícono que prefiera (algunos íconos de ejemplo son proporcionados en *Archivos de Programa\AutoIt3\Aut2Exe\Icons*).
6. Si usted no quiere que nadie pueda decompilar su script (siempre y cuando haya un decompilador disponible) entonces debería ingresar una clave en la entrada destinada a tal fin.
7. La única opción restante que usted quizás desee cambiar es el nivel de compresión (especialmente si usa FileInstall para agregar archivos extra). Use el menú de *Compresión* para ajustar ese valor a su gusto. Al igual que con todas las rutinas de compresión, los mejores niveles los conseguirá seleccionando la opción más lenta. Sin embargo, sin importar que nivel de compresión elija, la velocidad de descompresión (mientras el .exe está corriendo) no variará.
8. Clic en *Convert* para compilar el script.

Nota: Los scripts pueden ser compilados con la extensión *.a3x. Ellos deberían ejecutarse en la forma AutoIt.exe NombreArchivo.a3x. El *.a3x contiene el script propiamente dicho, además de los archivos incluidos con FileInstall. Este formato permite distribuir archivos más pequeños ya que no incluyen dentro del mismo script compilado el intérprete AutoIt3.exe. Usted necesitará disponer sin embargo del archivo AutoIt3.exe en la máquina destinataria, pero de ningún otro.

Método 2 - Clic derecho

Únicamente disponible si se realizó la instalación completa.

1. Con el Explorer navegue hasta el archivo .au3 que se dispone a compilar.
2. Clic en el archivo para desplegar el menú emergente.



3. Se efectuará la compilación del archivo seleccionado en segundo plano y con el mismo nombre de origen - exceptuando la extensión, que cambiará a .exe.

Cuando se compila de esta forma, Aut2Exe utiliza los valores de ícono/compresión actuales (desde la última vez que Aut2Exe fue ejecutado manualmente a través del método 1).

Método 3 - La Línea de comandos

El programa *Aut2Exe.exe* puede ejecutarse desde la línea de comandos como sigue:

```
Aut2exe.exe /in <ArchivoEntrada.au3> [/out <ArchivoSalida.exe>] [/icon <ArchivоВIcono.ico>]
[/nodecompile] [/comp 0-4] [/pass <clave>] [/nopack][x64] [/bin <binfile.bin>]
```

Donde

Parámetro	Uso	Valor predefinido
/in	<ArchivoIngresado.au3> Especifica la ruta y el nombre del archivo del script a compilar.	Ninguno. El archivo debe ser proporcionado
/out	<ArchivoDeSalida.exe> Especifica la ruta y el nombre del archivo compilado. <ArchivoDeSalida.a3x> Especifica la ruta y el nombre del archivo cuando crea uno del tipo *.a3x	El nombre del archivo ingresado con la extensión .exe
/icon	<ArchivoDelIcono.ico> Especifica la ruta y el nombre del archivo a ser usado por el exe compilado.	El ícono Autolt
/comp	Declara el nivel de compresión que será usado cuando codifique el script (esto NO tiene relación con el UPX). Debe ser un número en el rango de 0 (ninguno) y 4 (máximo).	2
/nopack	Especifica que el archivo no deberá ser comprimido con UPX después de la compilación.	comprimir
/pack	Especifica que el archivo debe ser comprimido con UPX después de la compilación.	comprimir
/x64	Especifica que el script deberá ser compilado para utilizarse en sistemas de arquitectura x64.	ver notas
/x86	Especifica que el script deberá ser compilado para utilizarse en sistemas de arquitectura x86 (32-bit).	ver notas
/console	Especifica que el script debe ser compilado como una aplicación de Consola.	aplicación Windows (/gui)

/gui	Especifica que el script debe ser compilado como una aplicación Windows.	aplicación Windows (/gui)
/bin	<ArchivoBin.bin> Especifica la ruta y nombre del archivo bin a ser usado para compilar el archivo.	busca en el directorio Aut2exe

Ejemplos de Línea de Comando

`/in c:\MiCodigo.au3 /out c:\MiEjecutable.exe /icon c:\IconoPropio.ico /x64`

Dará como resultado la creación de c:\MiEjecutable.exe con una compresión normal, un ícono personalizado y estará compilado para utilizarse en sistemas de arquitectura x64.

`/in c:\MiCodigo.au3`

Dará como resultado la creación de un ejecutable unicode c:\MiCodigo.exe con compresión normal, ícono predefinido de Autolt, y apto para correr en sistemas win_32.

Notas de la Línea de Comando

Los nombres de archivo que son largos deberían encerrarse entre comillas, como por ej. "C:\Archivos de Programa\Test\Prueba.au3".

Con la excepción de /in todos los demás parámetros son opciones.

Si no se especifica una "salida" para el archivo, se establecerá el mismo nombre que para la entrada, pero con la extensión .exe.

Por defecto, el compilador de 32-bit produce un binario de 32-bit y el compilador de 64-bit produce binarios de 64-bit. Puede utilizar los parámetros /x86 y /x64 para especificar explícitamente el tipo de salida.

Los parámetros /pass y /nodecompile son redundantes para la versión 3.2.8.1. Estos serán ignorados y descartados prontamente.

The /ansi and /unicode switches are redundant as of version 3.3.0.0.

Los caracteres /ansi y /unicode son redundante para la versión 3.3.0.0.

Detalles Técnicos

Los scripts compilados y aquellos archivos adicionales que fueran añadidos con FileInstall, son comprimidos con el esquema de (Jon).

Debido a que un script compilado tiene que disponer de la habilidad de "ejecutarse" sin requerir de una clave que le posibilite desencriptarse a sí mismo - ej., el encriptado es en *dos-sentidos*, usted debe considerar al exe compilado como una *codificación* cuyo proceso no puede ser señalado libre de todo riesgo. Podríamos sin embargo asegurarle que el usuario-final no tendrá un fácil acceso al contenido del archivo *.exe.

Herramienta de información de ventana de Autolt

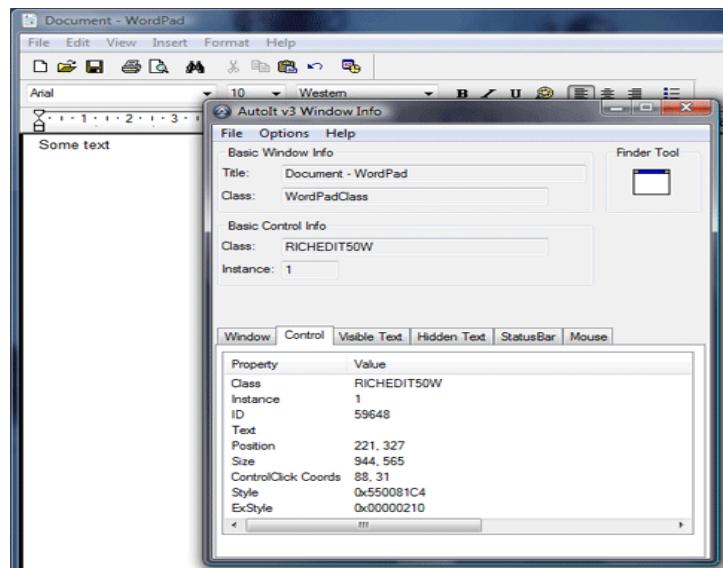
Autolt v3 viene con una herramienta integrada, **La Herramienta Autolt para Información de Ventanas** (Archivos de programa\Autolt3\AU3Info.exe). AU3Info le permite capturar información desde una ventana especificada, que puede utilizarse luego para automatizarla. La información obtenida incluye:

- [Títulos de ventanas](#)
- Texto de la ventana (visible y oculto)
- Tamaño y posición de la ventana
- Contenido de la barra de estado
- Ubicación del puntero del ratón
- Color de los pixels bajo el cursor
- Detalles del [Control](#) bajo el puntero del mouse.

Para usar AU3Info simplemente ejecútelo desde la línea de comandos, o bien el Menú Inicio. La ventana AU3Info permanecerá con atributo "siempre encima", de modo que siempre podrá leerla. Elija ahora la ventana en la que está interesado obtener información y actívela - el contenido de AU3Info cambiará entonces, y mostrará la información que esté disponible en ella. ¡Con la ayuda de AU3Info, antes que pueda darse cuenta ya estará automatizando ventanas!

Cuando AU3Info está en ejecución, si usted desea obtener una copia del texto, puede usar **CTRL-C** y pegarla directamente en su script, evitará así posibles errores al transcribir. Para pestañas que contienen información en una vista de lista (como la información del control mostrada debajo), simplemente haga **doble-click** en la entrada y la copiará al portapapeles. Sin embargo puede tornarse complicado si lo que usted quiere es capturar información de un pixel del ratón que está en cambio constante. Para facilitar las cosas puede "detener" la salida de datos de AU3Info presionando **CTRL-ALT-F**. Presione estas teclas nuevamente para "retomar".

Aquí hay un ejemplo de AU3Info sobre una Ventana del Editor "WordPad":



Títulos y texto de Ventanas (Básico)

Al automatizar, muchas ventanas pueden ser identificadas únicamente por su **título** o una combinación de su **título & texto**. Y utilizando la [Herramienta de Información de Ventana de AutoIt](#)(o a primera vista) esta información puede ser obtenida fácilmente. Los títulos de muchas ventanas son muy obvios, por ejemplo **Untitled - Notepad** es el título del programa notepad.exe y en muchos casos esto es suficiente para automatizar.

Nota: Si una cadena vacía "" es da para el **título** y el **texto** entonces la ventana *activa* puede ser usada (esto no es cierto en algunos de los más avanzados descritos en [WinTitleMatchModes](#))!

El título de ventana y el texto es **case sensitivo**. Usted debe hacer coincidir la fuente de letra y la puntuación exactamente! Para evitar problemas seleccione el título/texto en la Herramienta de Información de Ventana y use **ctrl-c** para copiar este y pegarlo directamente en su script.

La mayoría de las funciones de ventanas de AutoIt tienen parámetros para el título y el texto, aquí mostramos la función WinWaitActive. Esta función pausa la ejecución del script **mientras** la ventana especificada es restaurada y activada.

WinWaitActive "título", ["texto"], [espera]

título es el único parámetro requerido en esta función, los demás: **texto** y **espera** son opcionales. En algunas funciones el parámetro **texto** no es opcional, si usted no tiene el deseo de especificar algún texto solamente use "" (una cadena en blanco). Una cadena en blanco, o absolutamente nada, en el **texto** le dice a AutoIt que cualquier texto es válido.

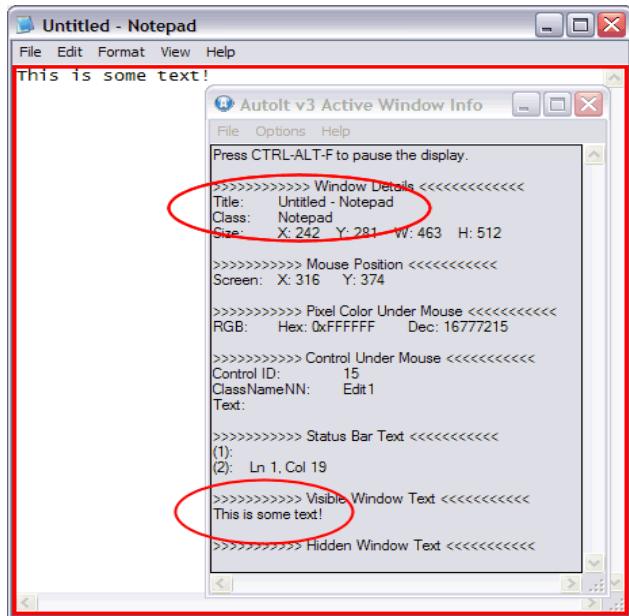
Para usar la función de arriba con cualquier ventana del notepad ambos métodos funcionan:

oAutoIt.WinWaitActive "Untitled - Notepad"

y

oAutoIt.WinWaitActive "Untitled - Notepad", ""

Si la misma ventana del notepad tiene "This is a line of text" tipeada dentro de la ventana, la [Herramienta de Información de Ventana de AutoIt](#) mostrará:



Note que la Herramienta de Información de Ventana a **visto** el título y el texto de la ventana del notepad. Cualquier cosa que viere la Herramienta de Información de Ventana lo puede ver también Autolt. Ahora poseemos información para identificar exactamente cada ventana del notepad abierta. En este caso procedemos así:

oAutoIt.WinWaitActive "Untitled - Notepad", "Esto es un texto!"

Texto de ventana

El texto de ventana consiste en todo el texto que Autolt puede "ver". Estos son usualmente algunas cosas como los contenidos de controles de edición (como el de más arriba "This is a line of text") pero también puede incluir otros texto como:

- Texto del botón como &Yes, &No, & Next (el signo & indica una letra subrayada)
- Texto de diálogos "Usted está seguro de estar comprendiendo?"
- Texto de controles
- Texto variado - algunas veces usted no sabe que significa :)

Lo importante es que usted puede utilizar el texto junto con el título para identificar únicamente una ventana y operar con esta.

Cuando usted especifica el parámetro **texto** en una función de ventana este es tratado como una cadena. De esta forma para el ejemplo usado más arriba "esto es un " usted obtendrá una coincidencia.

Lo que se ha descrito es el modo por defecto como opera Autolt, existen [modos avanzados](#) para usar cuando las cosas no son tan simples.

Nota: Ventanas ocultas pueden coincidir por "título" solo si "texto" está en blanco ("").

Títulos y texto de Ventanas (Avanzado)

Autolt opera en uno de los tres modos de "Coincidencia de Ventana". Estos modos son establecidos con la función [AutoItSetOption](#) usando la opción [WinTitleMatchMode](#).

Modo 1 (por defecto)

Coincidencia parcial del título desde el inicio.

En este modo el título de ventana **Untitled - Notepad** coincide con cualquiera de estos: "Untitled - Notepad", "Untitled", "Un", etc.

Ejemplo:

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

Modo 2

Coincidencia de cualquier sub-cadena en el título.

En este modo un título como **Untitled - Notepad** coincide con cualquiera de estos: "Untitled - Notepad", "Untitled", "Notepad", "pad", etc.

Ejemplo:

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

Modo 3

Coincidencia exacta del título.

En este modo un título de ventana como: **Untitled - Notepad** solo coincide con: "Untitled - Notepad"

Modo 4 (Combinando los modos anteriores)

Modo avanzado

Debe ser reemplazado con Descripción Avanzada de Ventanas pues no necesita ningún modo para ser establecido.

Modo -1 a -4

Fuerza la coincidencia en minúscula acorde con otro tipo de coincidencia.

Descripción Avanzada de Ventanas

Una descripción especial puede ser utilizada como el parámetro título de la ventana. Esta descripción puede ser usada para identificar una ventana por las siguientes propiedades:

- **TITLE** -Título de la ventana
- **CLASS** - El nombre de clase (classname) de la ventana
- **REGEXPTITLE** -Título de la ventana usando expresiones regulares
- **REGEXPCLASS** -El nombre de clase (classname) de la ventana usando expresiones regulares
- **LAST** - La última ventana usada en un comando previo de AutoIt
- **ACTIVE** - Ventana actualmente activa
- **X \ Y \ W \ H** - la posición y tamaño de la ventana
- **INSTANCE** - La intancia de base-1 cuando todas las propiedades coinciden.
- **HANDLE** - La dirección del identificador retornado por el método [WinGetHandle](#)

Una o más propiedades son usadas en el parámetro título parameter de un comando de ventana en el formato:

[PROPIEDAD1:Valor1; PROPIEDAD2:Valor2]

Nota : si un Valor contiene el carácter ";" este debe ser duplicado.

ejemplo: Espera una ventana de nombre de clase (classname) "Notepad"

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

ejemplo: Cierra la ventana activa actual

oAutoIt.WinClose "[ACTIVE]", ""

ejemplo: Espera por la una 2da instancia de la ventana con título: "Mi Ventana" y classname "My Class"

oAutoIt.WinWaitActive "[TITLE:My Window; CLASS:My Class; INSTANCE:2]", ""

Nota : si un valor debe contener ";" este debe ser duplicado.

Controles

Uno de los rasgos novedosos más destacables en Autolt v3 es la habilidad para trabajar directamente con diversos tipos de *Controles* en Ventanas. Casi todo lo que usted puede ver dentro de una Ventana es un control: botones, etiquetas de texto, grupos, campos de edición (por citar solo algunos) son controles. De hecho, el sencillo Bloc de Notas jes un enorme "control Editor" ! Como Autolt actúa directamente con los controles, es un método muy confiable para automatizar todo, en vez de simplemente efectuar los envíos desde el teclado.

Nota: *Autolt solamente funciona con controles estándar de Microsoft - algunas aplicaciones usan controles propios y aunque pueden lucir muy similares a un control de MS probablemente resistan la automatización. ¡Experimente!*

Usando la [Herramienta de Información de Ventana de Autolt](#) usted puede mover su ratón en torno al interior de la ventana que le interesa y podrá tomar información del control que en ese momento se encuentre bajo la posición del ratón.

- Control ID
- ClassNameNN
- Texto

Donde usted vea un comando [Control...\(\)](#) se espera un parámetro **Control ID** (en la mayoría de ellos). Usted puede usar **uno** de estos métodos. El método que usted escoja dependerá de sus preferencias personales y de la información que sea capaz de extraer de la Herramienta de Información de Ventana de Autolt. En general, el mejor método para usar es el **Control ID**, con los otros métodos se puede obtener resultados si Control ID no es disponible o no es único (**por lo general con texto estático cada parte del texto en realidad tiene el mismo ID de control por lo que tendrá que utilizar uno de los otros métodos para trabajar con los controles**)

Control ID

El ID del Control es un identificador numérico interno que Windows asigna a cada control. Podría decirse que es (por lo general) el mejor método para identificar controles. Además de la Herramienta Autolt para información de Ventanas, otras aplicaciones tales como lectores de pantalla para personas ciegas y herramientas de Microsoft/Apis pueden permitirle también acceder al ID del Control.

ClassNameNN

Cada control estándar de Microsoft tiene un "nombre de clase" como "botón" o "editor". En Autolt éste se combina con la asignación de una "instancia" propia para cada control, dado su método "NombreDeClaseNN". Si usted visualiza una ventana con unos pocos botones contenidos en la misma, a ellos se referirá entonces como "Botón1", "Botón2", "Botón3", etc.

Este método es muy útil cuando el ID del control no puede emplearse (como texto estático y controles personalizados).

Texto

Sepa a partir de ahora que AU3Info le proporciona el texto visible en un control, así por ejemplo un botón con la leyenda **Siguiente** podrá verlo con el texto **&Siguiente** - el & le indica que la letra consecutiva está subrayada, circunstancia común cuando trabaja con menús y controles. Usted podrá usar ese texto para identificar el control en lugar de utilizar el "NombreDeClaseNN", si así lo desea - Sin embargo, si muchos controles poseen el mismo texto, emplear éste método puede resultarle problemático.

Observe los contenidos de [Referencia de Método \ Administración de Ventanas\ Controles](#) para una lista de funciones que trabajan con estos controles.

Soporte Unicode

Desde la versión 3.2.4.0 Autolt es ofrecido como un programa Unicode. La versión Unicode permite que sus amigos del mundo puedan entender los caracteres y scripts de Autolt!

Nota: la versión Unicode de Autolt (Autolt3.exe) y los scripts compilados en modo Unicode solo pueden ser ejecutados bajo Windows NT/2000/XP/2003/Vista y máquinas posteriores. Para permitir a los scripts ejecutarse bajo Windows 9x estos deben usar las versiones anteriores de Autolt. La última versión compatible con Windows 9x fué la 3.2.12.x.

- Autolt3.exe - Versión Unicode
- Autolt3A.exe - Versión ANSI

¡Las versiones Unicode permitirán a nuestros amigos de todo el mundo finalmente usar Autolt y también scripts con caracteres extendidos!

Note: La versión Unicode de Autolt (Autolt3.exe) y los scripts compilados en modo Unicode serán solo ejecutados en Windows NT/2000/XP/2003/Vista y plataformas posteriores. Para permitir a scripts ejecutarse en Windows 9x estos deben diseñarse en versiones anteriores de Autolt. La última versión compatible con Windows 9x fué 3.2.12.x.

Autolt leerá un fichero de script en formato ANSI o UTF16 (big o little endian) / UTF8 con un [BOM](#) válido. En adición, funciones como [FileReadLine](#) automáticamente leerán texto en ficheros ANSI y UTF16/UTF8 proveyendo un [BOM](#) encontrado. Ficheros UTF8 con o sin BOM son también soportados.

Funciones de salida como [FileWriteLine](#) pueden usar los formatos ANSI, UTF16 y UTF8 - pero los ficheros deben ser habiertos en el modo deseado usando los flags adecuados en la función [FileOpen](#) de otro modo ANSI será el modo usado.

Los formatos de de ficheros de texto soportados y scripts en los editores populares son mostrados en esta tabla:

Notación de Autolt	Notepad	Notepad++	SciTe (Editor por defecto en Autolt)
ANSI	ANSI	ANSI	8 bit / Code Page Property
UTF16 LittleEndian	Unicode	UCS-2 LittleEndian	UCS-2 LittleEndian
UTF16 BigEndian	Unicode big endian	UCS-2 BigEndian	UCS-2 BigEndian
UTF8 with BOM	UTF-8	UTF-8	UTF-8 with BOM
UTF8 without BOM	Not supported	UTF-8 without BOM	UTF-8

El formato de script recomendado es UTF-8. Formatos ANSI no son recomendados para otros lenguajes aparte de Inglés pues puede provocar errores cuando se ejecutan en máquinas con otro entorno local.

Limitaciones actuales

Existen algunas porciones de Autolt que aún no soportan totalmente Unicode. Estas son:

- [Send](#) y [ControlSend](#) - en su lugar, Use las funciones [ControlSetText](#) o [Clipboard](#).
- Operaciones de [Console](#) son convertidos a ANSI.

Estas limitaciones serán atendidas en futuras versiones si es posible.

Corriendo bajo Edición de Windows de 64-bits

Autolt ha sido tradicionalmente una aplicación de 32-bits. Con el lanzamiento de la versión 3.2.10.0 algunos componentes nativos x64 fueron agregados, incluyendo:

- Autolt (Autolt3_x64.exe)
- Aut2Exe (Aut2Exe_x64.exe)
- Au3Info (Au3Info_x64.exe)
- AutoItX (AutoItX3_x64.dll)

Durante la instalación, si se está corriendo sobre una plataforma x64 se le dará opción de instalar y configurar la versión a x64. Esas versiones son compatibles al 100% sobre x64, sin embargo, algunos scripts que usan [DllCall/DllStruct](#) con estructuras personalizadas, pueden usar valores que causen conflictos con la compatibilidad a 64-bits (tales como usar enteros para punteros de 32-bits). Esto incluye algunos de los UDFs suministrados con Autolt, los cuales no han sido revisados bajo x64.

Puede correr la versión x86 de Autolt haciendo click derecho sobre un script dado y seleccione de la lista la opción "Ejecutar Script (x86)".

Si sospecha que un script no está trabajando correctamente bajo x64 pero si lo hace bajo x86, por favor, repórtelo como un bug.

Para ver si su Edición de Windows es una versión de 64-Bits, use la macro @OSArch. Para ver si se está utilizando la versión de Autolt a 32 o 64-bits, use @AutoItX64.

Corriendo la versión 32-bits de Autolt sobre un Sistema x64

Para acceder a archivos, Windows tiene un mecanismo de redirección especial hacia algunos directorios del sistema:

Directrios	Valores 32-bits	Valores 64-Bits
@SystemDir	@Windowsdir & "\System32"	@Windowsdir & "\SYSWOW64"
@ProgramFilesDir	{systemDrive} & "\Program Files"	{SystemDrive} & "\Program Files(x86)"

Es posible acceder a la versión de 64-bit de algunos directorios desactivando el mecanismo de redirección.

`DllCall("kernel32.dll", "int", "Wow64DisableWow64FsRedirection", "int", 1)`

Puede encontrar alguna información adicional en [MSDN](#).

Para el registro, use HKCR64 o HKLM64 si desea omitir el mecanismo de redirección, vea documentación sobre Funciones del Registro.

Referencia del Lenguaje - Tipos de Datos

En Autolt hay un único tipo de dato llamado **Variante** (Variant). Una variante puede contener datos numéricos o de cadena dependiendo de lo que se que almacene en ella y de la situación en que esta se emplee. Por ejemplo, si usted efectúa una multiplicación entre dos variantes, ellas entonces serán tratadas como números, si en cambio concadena (une) dos variantes serán tratadas como cadenas.

Algunos ejemplos:

*10 * 20 equivale al **número** 200 (* se utiliza para multiplicar entre dos números)*

*10 * "20" equivale al **número** 200*

*"10" * "20" equivale al **número** 200*

*10 & 20 equivale a la **cadena** "1020" (& se utiliza para unir dos cadenas diferentes)*

Si una cadena se utiliza como número, se hace un llamado implícito a la función [Number\(\)](#). Si ésta no contiene ningún número, entonces se asume un valor igual a 0. Por ejemplo,

*10 * "fgh" equivale al **número** 0.*

Si una cadena se utiliza como valor booleano y la misma es una cadena vacía "", será considerada como False (vea debajo). Por ejemplo,

*NOT "" equivale al booleano **true**.*

Números

Los números pueden ser decimales estándar tales como **2**, **4.566**, y **-7**.

También soporta la notación científica; por consiguiente, usted puede escribir **1.5e3** en lugar de **1500**.

Los números enteros pueden a su vez representarse con notación hexadecimal, si precediendo al entero usted coloca **0x** así **0x409** o **0x4fff** (cuando usa notación hexadecimal solamente números de 32-bit son válidos).

Cadenas

Las cadenas deben expresarse encerradas entre comillas como "estas". Si usted quiere usar una cadena que contiene comillas en su interior, duplíquelas:

"aquí está entre ""comillas-dobles"" - ¿se comprendió?"

Incluso puede usar apóstrofes como 'estos' y 'colocar también 'apóstrofes-dobles' - ¿si?'

Combine ambos métodos para facilitarse el trabajo y evitar el duplicado de comillas en forma sencilla. Por ejemplo, si lo que quiere es usar una enorme cantidad de comillas en cadenas, entonces utilice apóstrofes para declararlas:

'Esta "sentencia" contiene "cantidad" de "comillas" ¿no es cierto?'

sería más simple que:

"Esta ""sentencia"" contiene ""cantidad"" de ""comillas"" ¿no es cierto?"

Cuando se evalúan, las cadenas pueden manifestar substitución de variables Env o Var de acuerdo a la definición mediante la función [Opt\(\)](#).

Booleanos

Los Booleanos son valores **lógicos**. Solamente hay dos valores posibles: **true** y **false**.

Es posible utilizarlos en asignaciones de variables, junto con los operadores Booleanos **and**, **or** y **not**.

Ejemplos:

`$Boolean1 = true`

`$Boolean2 = false`

`$Boolean3 = $Boolean1 AND $Boolean2`

Resultará que \$Boolean3 es **false**

`$Boolean1 = false`

`$Boolean2 = not $Boolean1`

Resultará que \$Boolean2 es **true**

Si los valores Booleanos son utilizados junto con números, se aplican las siguientes reglas:

Un valor numérico 0 equivale al Booleano **false**

Cualquier otro número equivale al Booleano **true**

Ejemplo:

`$Number1 = 0`

`$Boolean1 = true`

`$Boolean2 = $Number1 and $Boolean1`

Resultará que \$Boolean2 es **false**

Si utiliza aritmética con los valores Booleanos (**¡lo cual no se recomienda en lo absoluto!**), se aplican las siguientes reglas:

Un Booleano true se convertirá en el valor numérico **1**

Un Booleano false se convertirá en el valor numérico **0**

Ejemplo:

```
$Boolean1 = true
```

```
$Number1 = 100
```

```
$Number2 = $Boolean1 + $Number1
```

Resultará que \$Number2 tiene por resultado 101

Si utiliza cadenas con los valores Booleanos, serán convertidos como sigue:

Un Booleano true será convertido al valor de cadena "**True**"

Un Booleano false será convertido al valor de cadena "**False**"

Ejemplo:

```
$Boolean1=true
```

```
$String1="La verdadera prueba es: "
```

```
$String2=$String1 & $Boolean1
```

Resultará que \$String2 es una cadena con el contenido "La verdadera prueba es: True"

Al revés sin embargo es diferente. Cuando usted usa comparaciones de cadenas con valores Booleanos, se aplican las siguientes reglas:

Únicamente una cadena vacía ("") equivale al Booleano **false**

Cualquier otro valor de cadena (incluso conteniendo el texto "0") será un Booleano **true**

Tipo Binario (Binary)

El tipo Binario puede almacenar cualquier valor de byte. Los mismos pueden ser convertidos a una representación hexadecimal cuando se almacenan en una variable del tipo Cadena.

Ejemplo:

```
$bin = Binary("abc")
```

```
$str = String($bin) ; "0x616263"
```

Puntero

El tipo Puntero almacena una dirección de memoria que puede ser de 32bits o 64bits dependiendo sobre que versión de AutoIt de 32bit o 64 bit se está trabajando. Son convertidos a una representación hexadecimal cuando se almacenan en una variable del tipo Cadena. Los punteros de ventana (HWND) son retornados con [WinGetHandle](#) como tipos de dato "Puntero".

Tipos de Datos y Rangos

La tabla siguiente muestra los tipos de datos variantes y sus respectivos rangos.

Ejemplo:

```
$bin = "abc" & chr(0) & "def"  
$bin = BinaryString("abc")
```

Rangos del Tipo de Datos

La tabla siguiente le muestra el rango de valores que un tipo de dato variante puede contener.

Sub-Tipo de Dato	Rango y Notas
Int32	Un número entero con signo a 32bit.
Int64	Un número entero con signo a 64bit.
Double	Un número de "doble precisión" de coma flotante.
String	Puede contener cadenas de hasta 2.147.483.647 caracteres.
Binary	Datos Binarios, pueden contener hasta 2.147.483.647 bytes.
Pointer	Un puntero a una dirección de memoria. De 32bit o 64bit dependiendo de la versión de Autolt que se utiliza.

Algunas funciones de Autolt solo trabajan con números de 32 bit (ej: [BitAND](#)) y son convertidos automáticamente - las funciones que lo hacen se documentan.

Referencia del lenguaje - Variables

Una variable es sencillamente, un lugar en la memoria donde se almacenan datos a los que es posible acceder rápidamente. Usted puede colocar alguna información allí y también puede recuperarla o cambiarla. Para exemplificar, usted debe crear una variable para guardar el texto con el que un usuario responde a una pregunta, o el resultado de una ecuación matemática.

Cada variable tiene un nombre que debe comenzar con el carácter \$ y sólo es posible que contenga **letras, números** y el carácter _ (guión bajo). Aquí algunos ejemplos de nombres:

*\$var1
\$mi_variable
\$_mivariable*

Cualquier variable es almacenada con el tipo de dato variante.

Declaración de variables

Las variables son declaradas y creadas con las palabras claves Dim, Local y Global :

Dim \$var1

O pueden declararse varias simultáneamente:

Dim \$var1, \$mivariable

Puede asignarle un valor a la variable **sin** haberla declarado primero, pero las declaraciones explícitas son preferibles en muchos casos.

\$var1 = "variable creada y asignada"

Declaración de Constantes

Las Constantes son declaradas y creadas con la palabra clave Const de este modo:

Const \$valorPI = 3.14, \$apostoles=12

Las Constantes pueden ser declaradas e inicializadas con la palabra clave Enum así:

*Enum \$const1 = 1, \$const2, \$const3 ; 1, 2, 3
Enum STEP 2 \$incr0, \$incr2, \$incr4 ; 0, 2, 4
Enum STEP *2 \$mult1, \$mult2, \$mult4 ; 1, 2, 4*

Las Constantes no pueden redeclarar una variable ya existente.

Entorno

El entorno de una variable es definido de acuerdo a donde y cuando se declara a la misma. Si usted declara la variable al inicio de su script y por fuera de cualquier función esta existirá en un entorno **Global** y podrá ser leída o cambiada en cualquier sección del script.

Si en cambio se declara *dentro* de una función el entorno será **Local** y sólo podrá ser usada *dentro de la misma función*. Las Variables creadas en el interior de una función son destruidas inmediatamente cuando esta finaliza.

De manera predefinida, cuando las variables son declaradas empleando Dim o se les asigna valor en una función, el entorno será **Local**, a excepción de que **haya** una variable Global con el mismo nombre (en cuyo caso la variable Global es reusada). Esto puede cambiarse al utilizar las palabras claves Local y Global para declarar variables y así **forzar** el entorno que usted desea para ellas.

Arreglos

Un Arreglo es una variable conteniendo elementos en una serie con datos del mismo tipo y tamaño. Es posible acceder a cualquier elemento de la variable mediante un índice numérico.

Ejemplo:

Digamos que usted quiere almacenar esta serie de caracteres: "A", "U", "T", "O", "I", "T" y "3". Usted puede emplear siete variables separadas para hacer eso, pero usando un arreglo sería mucho más eficiente:

```
$Arreglo[1]="A"  
$Arreglo[2]="U"  
.etc..  
$Arreglo[7]="3"
```

La estructura del arreglo \$Arreglo puede verse de esta forma:

\$Arreglo	1	2	3	4	5	6	7	<=índice
	A	U	T	O	I	T	3	<= valor

Cada valor dentro del arreglo posee un índice de referencia, por ejemplo, el valor 'O' tiene el índice 4.

Para acceder a un valor específico del arreglo, usted sólo debe saber el número índice asignado:

```
$MyChar = $Arreglo[4]
```

Resultará que \$MyChar contiene la letra "O" (Vea también: 'operadores').

El índice numérico puede también sustituirse con otra variable o una expresión, por lo tanto usted puede constituir complejas formas para asignar o acceder a elementos en un arreglo.

Los Arreglos pueden crearse multi-dimensionales, cuando usted utiliza múltiple series de índices numéricos, como ser:

```
$Arreglo[1][1]="Arriba-Izq"  
$Arreglo[2][1]="Abajo-Izq"  
$Arreglo[1][2]="Arriba-Derecha"  
$Arreglo[2][2]="Abajo-Derecha"
```

(Estos valores son simplemente para ejemplificar)

Usted puede usar un máximo de 64 dimensiones en un arreglo. El número total de entradas no puede ser mayor que 2^{24} (16 777 216).

Antes de comenzar a utilizar un arreglo en su script, debe definir el límite del mismo empleando la palabra clave 'Dim'.

Tipos de datos en Arreglos

Nosotros mencionamos que un Arreglo puede contener sólo un tipo de datos. Pero técnicamente hablando, una Variante en Autolt puede contener cualquiera, desde un número hasta un valor booleano. Por lo tanto, un arreglo-Autolt tiene la capacidad de almacenar cualquier tipo, incluso otros arreglos:

```
$Arreglo[1]=1  
$Arreglo[2]=true  
$Arreglo[3]="Texto"  
$Arreglo[4]=$OtroArreglo
```

Esto no ha sido estrictamente prohibido en Autolt. Sin embargo, NO ES RECOMENDABLE mezclar diferentes tipos de datos. Especialmente el uso de un Arreglo dentro de otro arreglo puede afectar severamente la velocidad de ejecución de su script.

Referencia del Lenguaje - Macros

Autolt tiene un número de Macros que son variables especiales de sólo-lectura. Las macros comienzan con el carácter @ en lugar del acostumbrado signo \$ así que son fácilmente distinguibles. Del mismo modo que con las variables, es posible usar macros en expresiones, sin embargo no puede asignarles un valor a ellas.

Las macros pre-definidas son generalmente utilizadas para proveer un acceso sencillo a información del sistema, como la localización del directorio Windows, o el nombre del usuario que ingresó al sistema.

Vaya por aquí para obtener una lista completa.

Macro	Descripción
@AppDataCommonDir	ruta a la carpeta común de Datos de aplicación
@AppDataDir	ruta de la carpeta de Datos de Aplicación del usuario actual
@AutoItExe	La dirección completa y nombre de archivo del archivo ejecutable AutoIt actualmente en ejecución. Para archivos compilados es la ruta completa del script compilado.
@AutoItPID	PID del proceso ejecutando el script.
@AutoItVersion	Número de Versión de AutoIt tal como 3.2.1.0
@AutoItX64	Devuelve 1 si el script está ejecutándose bajo la versión nativa de x64 de AutoIt.
@COM_EventObj	Objeto del evento COM está siendo notificado. Válido solamente en la función de eventos COM.
@CommonFilesDir	ruta a la carpeta de Archivos Comunes (Common Files)
@Compiled	Devuelve 1 si el script es ejecutable compilado; de otra manera, devuelve 0.
@ComputerName	Nombre de la Computadora en la red.
@ComSpec	valor de %comspec%, el interpretador secundario especificado de Comandos; primordialmente para usos de línea de comando, ejemplo <i>Run(@ComSpec & "/k help more")</i>
@CPUArch	devuelve "X86" cuando la CPU es una CPU de 32-bit y "X64" cuando la CPU es de 64-bit.

@CR	Retorno de Carro, Chr(13); algunas veces es usado para interrupción de línea.
@CRLF	= @CR & @LF ;ocasionalmente usado para interrupción de líneas.
@DesktopCommonDir	ruta a la carpeta de Escritorio (Desktop)
@DesktopDir	ruta a la carpeta del Escritorio del usuario
@DesktopHeight	Altura de la pantalla del escritorio en píxeles. (resolución vertical)
@DesktopWidth	Ancho de la pantalla del escritorio en píxeles. (horizontal resolución)
@DesktopDepth	Rango de la pantalla del escritorio en bits por píxel.
@DesktopRefresh	Tasa de refresco de la pantalla del escritorio en hercios.
@DocumentsCommonDir	ruta de la carpeta a Documentos
@error	Bandera de estado de error. Ver función SetError.
@exitCode	Código de Salida establecido por el enunciado Exit.
@exitMethod	Método de salida. Ver Func OnAutoltExitRegister().
@extended	Valor devuelto por la función Extended - usado en ciertas funciones tal como StringReplace.
@FavoritesCommonDir	ruta a Favoritos
@FavoritesDir	ruta a la carpeta de Favoritos del usuario actual
@GUI_CtrlId	Identificador del último control pulsado. Válido solamente en una función de evento. Ver la función GUICtrlSetOnEvent.
@GUI_CtrlHandle	Identificador del último handle pulsado. Válido solamente en una función de evento. Ver la función GUICtrlSetOnEvent.
@GUI_DragId	Identificador del último control arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_DragFile	Nombre del archivo que está siendo arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.

@GUI_DropId	Identificador del control arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_WinHandle	El handle de la última ventana GUI pulsada. Válido solamente en Función de evento. Ver la función GUICtrlSetOnEvent.
@HomeDrive	Letra de la unidad conteniendo al directorio de inicio del usuario.
@HomePath	Parte del Directorio de Inicio del usuario actual. Para obtener la dirección completa, usarlo en conjunto con @HomeDrive.
@HomeShare	Servidor y nombre de recurso compartido del directorio inicio del usuario actual.
@HOUR	Valor de Horas del reloj en formato de 24-horas. Rango es 00 a 23
@HotKeyPressed	Último hotkey presionado. Ver la función HotKeySet.
@IPAddress1	Dirección IP del primer adaptador de red. Tiende a devolver 127.0.0.1 en algunas computadoras.
@IPAddress2	Dirección IP del segundo adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@IPAddress3	Dirección IP del tercer adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@IPAddress4	Dirección IP del cuarto adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@KBLLayout	Devuelve código de acuerdo al esquema del Teclado. Ver Apéndice para posibles valores.
@LF	Avance de línea, Chr(10); típicamente usado para saltos de línea.
@LogonDNSDomain	Inicio de sesión de dominio de DNS.
@LogonDomain	Inicio de sesión de dominio.
@LogonServer	Inicio de sesión de servidor.
@MDAY	Día del mes actual. Rango es de 01 a 31
@MIN	Valor de los minutos del reloj. Rango es de 00 a 59

@MON	Mes en curso. Rango es de 01 a 12
@MSEC	Valores de Milisegundos de reloj. Rango entre 00 y 999
@MUILang	Retorna el código multilenguaje si es disponible (En Vista funciona por defecto). Ver Apéndice para posibles valores.
@MyDocumentsDir	ruta de la carpeta Mis Documentos
@NumParams	Número de parámetros utilizados para llamar a los usuarios funciones
@OSArch	Devuelve un valor de los siguientes: "X86", "IA64", "X64" - Esto es el tipo de arquitectura del sistema operativo actual en ejecución.
@OSBuild	Devuelve el número de build del Sistema Operativo. Por ejemplo, Windows 2003 Server devuelve 3790
@OSLang	Devuelve código de acuerdo al lenguaje del Sistema Operativo. Ver Apéndice para valores posibles.
@OSServicePack	Información de Service pack en la forma de "Service Pack 3"
@OSTYPE	Devuelve "WIN32_NT" para NT/2000/XP/2003/Vista/2008/Win7/2008R2.
@OSVersion	Devuelve uno de los siguientes: "WIN_2008R2", "WIN_7", "WIN_2008", "WIN_VISTA", "WIN_2003", "WIN_XP", "WIN_XPe", "WIN_2000".
@ProcessorArch	Devuelve un valor de los siguientes: "WIN_2008", "WIN_VISTA", "WIN_2003", "WIN_XP", "WIN_2000"
@ProgramFilesDir	ruta a la carpeta de Archivos de Programa
@ProgramsCommonDir	ruta a la carpeta del Menú de Inicio
@ProgramsDir	ruta a la carpeta de Programas del usuario actual (carpeta de Menú de Inicio)
@ScriptDir	Directorio contenido el script ejecutándose. (El resultado no contiene una pleca al final)
@ScriptFullPath	Equivalente a @ScriptDir & "\" & @ScriptName
@ScriptLineNumber	Número de línea que está siendo ejecutada en el script actual. Útil para enunciados de depuración especialmente cuando una función es llamada para evaluarse. (No es significante en un

	script compilado)
@ScriptName	Nombre largo del archivo script en ejecución.
@SEC	Valor en segundos de la hora. Rango es 00 a 59
@StartMenuCommonDir	ruta de acceso para el menú Inicio común
@StartMenuDir	ruta de acceso para el menú Inicio del usuario actual
@StartupCommonDir	ruta a la carpeta de Inicio
@StartupDir	carpeta del inicio del usuario actual
@SW_DISABLE	Desactiva la ventana.
@SW_ENABLE	Activa la ventana.
@SW_HIDE	Oculta la ventana y activa otra.
@SW_LOCK	Bloquea la ventana evitando así pintar sobre ella.
@SW_MAXIMIZE	Maximiza la ventana especificada.
@SW_MINIMIZE	Minimiza la ventana especificada y activa la próxima al frente según el orden en Z.
@SW_RESTORE	Activa y muestra la ventana. Si la ventana está minimizada o maximizada, el sistema restaura a su posición y tamaño original. Una aplicación debería especificar esta bandera cuando se restaura una ventana minimizada.
@SW_SHOW	Activa la ventana y muestra en su posición y tamaño actual.
@SW_SHOWDEFAULT	Establece el estado de muestra de una ventana basado en el valor SW_ especificado por el programa que inició la aplicación.
@SW_SHOWMAXIMIZED	Activa la ventana y la muestra como ventana maximizada.
@SW_SHOWMINIMIZED	Activa la ventana y la muestra como ventana minimizada.
@SW_SHOWMINNOACTIVE	Muestra la ventana como ventana minimizada. Este valor es similar a @SW_SHOWMINIMIZED, excepto que la ventana no es activada.
@SW_SHOWNA	Muestra la ventana en su tamaño actual y posición. Este valor es similar a @SW_SHOW, excepto que la ventana no es activada.

@SW_SHOWNOACTIVATE	Muestra una ventana en su más reciente posición y tamaño. Este valor es similar a @SW_SHOWNORMAL, excepto que la ventana no es activada.
@SW_SHOWNORMAL	Activa y muestra una ventana. Si la ventana está minimizada o maximizada, el sistema restaura su posición y tamaño original. Una aplicación debería especificar esta bandera cuando está mostrando la ventana por primera vez.
@SW_UNLOCK	Desbloquea una ventana bloqueada para remuestreo.
@SystemDir	ruta a la carpeta de Windows' System (o el System32)
@TAB	carácter Tab, Chr(9)
@TempDir	Ruta a la carpeta de archivos temporales.
@TRAY_ID	Identificador del último ítem pulsado durante una acción en un TraySet(Item) OnEvent.
@TrayIconFlashing	Devuelve 1 si el ícono de bandeja está parpadeando; de otra manera, devuelve 0.
@TrayIconVisible	Devuelve 1 si el ícono de bandeja está visible; de otra manera, devuelve 0.
@UserProfileDir	ruta a la carpeta del perfil del usuario actual.
@UserName	ID del usuario actual que ha iniciado sesión.
@WDAY	Numérico el día de la semana. La escala es de 1 a 7, que corresponde de domingo a sábado.
@WindowsDir	ruta a la carpeta de Windows
@WorkingDir	Directorio actual de trabajo activo. (Los resultados no contiene una barra invertida)
@YDAY	Día actual del año. El rango es de 1 a 366 (o 001 a 365 si no es un año bisiesto)
@YEAR	Año actual en cuatro-dígitos

Referencia del Lenguaje - Operadores

Autolt tiene las siguientes asignaciones para operadores matemáticos, comparativos y lógicos.

Operador	Descripción
Operadores de Asignación	
=	Asignación. Ej: \$var = 5 (<i>asigna el número 5 a \$var</i>)
+=	Asignación incremental. Ej: \$var += 1 (<i>incremento de 1 a \$var</i>)
-=	Asignación decremental.
*=	Asignación multiplicadora.
/=	Asignación divisora.
&=	Asignación concatenadora. Ej: \$var = "uno" , y entonces \$var &= 10 (<i>ahora \$var es igual a "uno10"</i>)
Operadores Matemáticos	
+	Suma dos números. Ej: 10 + 20 (<i>es igual a 30</i>)
-	Resta dos números. Ej: 20 - 10 (<i>es igual a 10</i>)
*	Multiplica dos números. Ej: 20 * 10 (<i>es igual a 200</i>)
/	Divide dos números. Ej: 20 / 10 (<i>es igual a 2</i>)
&	Concadena/une dos cadenas. Ej: "uno" & 10 (<i>es igual a "uno10"</i>)
^	Eleva un número a la potencia. Ej: 2 ^ 4 (<i>es igual a 16</i>)
Operadores de Comparación (caso no sensitivo usado con cadenas excepto para ==)	
=	Comprueba si dos valores son iguales (insensible a mayúsculas/minúsculas para cadenas). Ej: If \$var= 5 Then (<i>verdadero si \$var almacena el valor 5</i>)
==	Comprueba si dos cadenas son iguales (sensible a mayúsculas/minúsculas para cadenas). Los operandos a ambos lados del operador son convertidos a cadena, si no lo son. Este operador solo debe ser utilizados para comparación de cadenas en modo caso sensitivo (sensible a mayúscula y minúsculas).

<>	Comprueba si dos valores son diferentes.(insensible a mayúsculas/minúsculas para cadenas). Para realizar una comparación de desigualdad sensible a mayúsculas/minúsculas en cadenas use: Not ("cadena1" == "cadena2")
>	Comprueba si el primer valor es mayor que el segundo. Las cadenas son comparadas lexicográficamente siempre aún si el contenido de la cadena es numérico.
>=	Comprueba si el primer valor es mayor o igual que el segundo. Las cadenas son comparadas lexicográficamente siempre aún si el contenido de la cadena es numérico.
<	Comprueba si el primer valor es menor que el segundo. Las cadenas son comparadas lexicográficamente siempre aún si el contenido de la cadena es numérico.
<=	Comprueba si el primer valor es menor o igual que el segundo. Las cadenas son comparadas lexicográficamente siempre aún si el contenido de la cadena es numérico.

Operadores Lógicos

AND	Operación lógica AND. Ej: If \$var = 5 AND \$var2 > 6 Then (<i>True si \$var es igual a 5 y \$var2 es mayor que 6</i>)
OR	Operación lógica OR. Ej: If \$var = 5 OR \$var2 > 6 Then (<i>True si \$var es igual a 5 o \$var2 es mayor que 6</i>)
NOT	Operación lógica NOT. Ej: NOT 1 (<i>es False</i>)

Cuando en una expresión se utiliza más de un operador, el orden secuencial de los sucesos es controlado por la **prioridad de los operadores**. La prioridad utilizada en Autolt es presentada debajo. Si dos operadores tienen la misma prioridad la expresión es evaluada de *izquierda a derecha*.

Desde la prioridad máxima a la mínima:

NOT

^

*** /**

+ -

&

< > <= >= = <> ==

AND OR

Ej: **2 + 4 * 10** es evaluado como **42**:

4 * 10 (equivale a 40)

$2 + 40$ (equivale 42)

Como el * tiene prioridad más alta que el + entonces ocurre **antes** que la adición.

$2 + 40 - 10$ (equivale a 32)

Primero se suma 2+40 y después se le resta 10

Usted puede usar paréntesis para forzar que una parte de la expresión sea evaluada primero.

Ej: **(2 + 4) * 10** equivale a **60**.

Nótese lo siguiente: cuando usa los operadores lógicos **AND**, **OR**:

Ej: **If MyFunc1() OR MyFunc2() Then**

(MyFunc2() no será llamada si MyFunc1() retorna true)

(MyFunc1() y MyFunc2() serán llamadas si MyFunc1() retorna False)

Ej: **If MyFunc1() AND MyFunc2() Then**

(MyFunc2() no será llamada cuando MyFunc1() retorna false)

(MyFunc1() y MyFunc2() serán llamada cuando MyFunc1() retorna True)

Referencia del Lenguaje - Declaraciones Condicionales

Usted frecuentemente deseará modificar el flujo de su Script, a través de una *condición* o serie de condiciones. ¿Es un número más grande que otro? O quizás: ¿esta cadena contiene cierto valor?

Las condiciones son evaluadas como **verdadero** (distinto de cero) o **falso** (cero). Por lo general se utilizan operadores como ==, <>, >= para realizar comparaciones.

Las siguientes declaraciones condicionales están disponibles en AutoIt:

If...Then...Else

Select...Case

Switch...Case

Todas estas declaraciones son similares y deciden que código se ejecutará dependiendo de las condiciones establecidas. Aquí hay un ejemplo de una declaración If que despliega un cuadro de mensaje en dependencia del valor de una variable:

```
$var = -20  
If $var > 0 Then  
    MsgBox(0, "Ejemplo", "$var es positivo!")  
Elseif $var < 0 Then  
    MsgBox(0, "Ejemplo", "$var es negativo!")  
Else  
    MsgBox(0, "Ejemplo", "$var es cero.")  
EndIf
```

En el ejemplo de arriba, la expresión **\$var > 0** es evaluada a **false** debido a que el valor es menor que cero. Cuando la primera condición falla, la segunda condición es testeada. La expresión **\$var < 0** es evaluada a **true**. Esto causa que la declaración siguiente se ejecute y muestre el segundo MsgBox y muestre: "\$var es negativo!"

Una declaración Select es muy parecida, pero se usa generalmente para situaciones donde usted quiere evaluar un gran número de condiciones, resultará más sencillo para leer entonces, que un extenso bloque del tipo If/Elseif. e.j:

```
$var = 30  
Select  
    Case $var > 1 AND $var <= 10  
        MsgBox(0, "Ejemplo", "$var es mayor que 1")  
    Case $var > 10 AND $var <= 20  
        MsgBox(0, "Ejemplo", "$var es mayor que 10")  
    Case $var > 20 AND $var <= 30  
        MsgBox(0, "Ejemplo", "$var es mayor que 20")  
    Case $var > 30 AND $var <= 40  
        MsgBox(0, "Ejemplo", "$var es mayor que 30")  
    Case $var > 40
```

```
    MsgBox(0, "Ejemplo", "$var es mayor que 40")
```

```
EndSelect
```

La declaración Switch también resulta similar a la declaración Select, pero por lo general se usa para situaciones donde la misma expresión es evaluada en comparación con algunos diferentes valores posibles.

```
$var = 30
```

```
Switch Int($var)
```

```
Case 1 To 10
```

```
    MsgBox(0, "Ejemplo", "$var es mayor que 1")
```

```
Case 11 To 20
```

```
    MsgBox(0, "Ejemplo", "$var es mayor que 10")
```

```
Case 21 To 30
```

```
    MsgBox(0, "Ejemplo", "$var es mayor que 20")
```

```
Case 31 To 40
```

```
    MsgBox(0, "Ejemplo", "$var es mayor que 30")
```

```
Case Else
```

```
    MsgBox(0, "Ejemplo", "$var es mayor que 40 o menor/igual a 0")
```

```
EndSwitch
```

En cada una de estas estructuras, el bloque de código de la primera condición evaluada a true es ejecutado. Todas las condiciones que siguen son ignoradas.

Referencia del Lenguaje - Declaraciones de Bucles

Un **bucle** o ciclo es una sección del código que se repite un determinado número de veces. Usted puede ejecutar un determinado bloque encerrándolo en un bucle, por ejemplo a la espera de que una determinada condición se cumpla, o sea, cambie a true o false.

Las siguientes declaraciones de Bucles están disponibles en Autolt:

For...Next

While...WEnd

Do...Until

For...In...Next

Referencia Keywords

For...To...Step...Next

Ciclo basado en una expresión.

**For <variable> = <inicio> To <parada> [Step <incremento>]
declaración**

...

Next

Parámetros

variable	variable usada para el contador.
inicio	El valor inicial de la variable.
parada	El valor final de la variable.
incremento	[opcional] El valor numérico (puede ser fraccionario) por el cual el contador es incrementado en cada ciclo. Por defecto es 1.

Comentarios

La variable será creada automáticamente con entorno LOCAL, aun cuando MustDeclareVars sea usado.

Declaraciones For...Next pueden ser anidadas. El ciclo FOR termina cuando el valor de la variable excede el valor de stop. Si incremento o parada es una variable, su valor es leído solamente la primera vez que el ciclo es ejecutado.

Un ciclo for es ejecutado cero veces si:

inicio > parada y step > 0, o

inicio < parada y step es < 0

Relativo

[ContinueLoop](#), [ExitLoop](#)

Ejemplo

For \$i = 5 to 1 Step -1

```
    MsgBox(0, "Cuenta regresiva!", $i)
```

Next

```
MsgBox(0, "", "Gran Suceso!")
```

While...WEnd

Ciclo basado en una expresión.

While <expresión>

Declaración

...

WEnd

Parámetros

expresión	Si la expresión es verdadera, toda la declaración hasta la declaración WEnd es ejecutado. El ciclo continua indefinidamente mientras la expresión sea verdadera.
-----------	--

Comentarios

Las declaraciones While...WEnd pueden ser anidadas. La expresión es testeada antes que el ciclo sea ejecutado de esta manera el ciclo es ejecutado cero o más veces. Para crear un ciclo infinito, usted puede usar un número diferente de cero en *expresión*.

Relativo

[ContinueLoop](#), [ExitLoop](#)

Ejemplo

\$i = 0

While \$i <= 10

```
    MsgBox(0, "Value de $i es:", $i)
```

```
    $i = $i + 1
```

WEnd

Do...Until

Ciclo basado en una expresión.

Do
declaración

...

Until <expresión>

Parámetros

expresión	La declaración entre Do y Until son ejecutado mientras la expresión sea verdadera.
-----------	--

Comentarios

La declaración Do...Until pueden ser anidadas.

La expresión es testeada después que el ciclo es ejecutado, de este modo el ciclo será ejecutada uno o más veces.

Relativo

[ContinueLoop](#), [ExitLoop](#)

Ejemplo

`$i = 0`

`Do`

`MsgBox(0, "Valor de $i es:", $i)`

`$i = $i + 1`

`Until $i = 10`

For...In...Next

Enumera elementos en un objeto de colección en un arreglo.

For <\$Variable> In <expresión>

Declaración

...

Next

Parámetros

Variable	Una variable para la cual un elemento es asignado.
----------	--

expresión	Debe ser una expresión que represente un Objeto, o un arreglo con al menos un elemento.
-----------	---

Comentarios

La variable será creada automáticamente con entorno LOCAL, aun cuando MustDeclareVars sea usado.

Si la expresión es un objeto de colección sin elementos o un arreglo multidimensional, el ciclo será obviado y la variable contendrá una cadena vacía.

Si la expresión no es un Object o un Arreglo, el script se detiene con un error, a menos que un identificador COM Error sea configurado.

Los arreglos en AutoIt son de solo lectura cuando utilizamos For...In. Mientras usted puede asignar un valor a la variable dentro del ciclo For...In, este valor no se refleja en el arreglo en sí mismo. Para modificar el contenido de un arreglo durante la enumeración use un ciclo For...To.

Declaraciones For...In...Next pueden ser anidadas.

Relativo

[With...EndWith](#)

Ejemplo

```
;Usando un arreglo
Dim $aArray[4]
$aArray[0] = "a"
$aArray[1] = 0
$aArray[2] = 1.3434
$aArray[3] = "test"

$string = ""
FOR $element IN $aArray
    $string = $string & $element & @CRLF
NEXT

Msgbox(0,"For..IN Arraytest","Resultado es: " & @CRLF & $string)

;Usando un objeto de colección
$oShell = ObjCreate("shell.application")
$oShellWindows = $oShell.windows
if Isobj($oShellWindows) then
```

```
$string=""  
for $Window in $oShellWindows  
    $String = $String & $Window.LocationName & @CRLF  
Next  
msgbox(0,"","Usted tiene las siguientes ventanas abiertas:" & @CRLF & $String)  
else  
    msgbox(0,"","Usted no ha abierto ninguna ventana.")  
endif
```

Referencia del Lenguaje - Declaraciones Obj

Un **obj** es la forma en como usted se refiere a un objeto. Usted quizás le interese para enumerar los elementos en un Objeto de Colección.

Las siguientes declaraciones de obj están disponibles en Autolt (Vistas anteriormente):

- [With...Endwith](#)
- [For...In...Next](#)

Se detallan más adelante...

Referencia del Lenguaje - Funciones del Usuario

Una función es una sección de código que puede llamarse en el Script para efectuar cierta "función" (valga la redundancia). Hay dos clases de funciones en Autolt, **funciones incorporadas propias del lenguaje y funciones del usuario (UDFs)**.

Funciones Incorporadas

La lista completa de funciones incorporadas propias del lenguaje se encuentra [aquí](#) y las notas sobre el empleo de las mismas figura [aquí](#).

Abajo se muestra una lista completa de las Funciones disponibles en Autolt. Click en el nombre de una función para más detalles de su descripción.

Función	Descripción
Abs	Calcula el valor absoluto de un número.
ACos	Calcula el arco coseno de un número.
AdlibRegister	Registra una función Adlib.
AdlibUnRegister	Des-registra una función adlib.
Asc	Devuelve el código ASCII de un carácter.
AscW	Devuelve el código unicode de un carácter.
ASin	Calcula el arco seno de un número.
Assign	Asigna un dato a una variable por su nombre.
ATan	Calcula el arco tangente de un número.

<u>AutoItSetOption</u>	Cambia el modo de operación de varias funciones/parámetros de AutoIt.
<u>AutoItWinGetTitle</u>	Devuelve el título de la ventana de AutoIt.
<u>AutoItWinSetTitle</u>	Cambia el título de la ventana de AutoIt.
<u>Beep</u>	Emite un sonido por el altavoz del sistema para el usuario.
<u>Binary</u>	Devuelve la representación binaria de una expresión.
<u>BinaryLen</u>	Devuelve el número de bytes de un dato variant binario.
<u>BinaryMid</u>	Extrae un número de bytes de un dato variant binario.
<u>BinaryToString</u>	Convierte un dato variant binario en una cadena.
<u>BitAND</u>	Realiza una operación AND a nivel de bit.
<u>BitNOT</u>	Realiza una operación NOT a nivel de bit.
<u>BitOR</u>	Realiza una operación OR a nivel de bit.
<u>BitRotate</u>	Realiza una operación de alternación de bit, con rotación.
<u>BitShift</u>	Realiza una operación de alternación de bit.
<u>BitXOR</u>	Realiza una operación OR exclusiva (XOR) a nivel de bit.
<u>BlockInput</u>	Habilita/deshabilita el mouse y el teclado.
<u>Break</u>	Habilita o desactiva la posibilidad del usuario de finalizar el script desde el ícono de la barra de tarea.
<u>Call</u>	Invoca una Función de usuario contenida en una cadena como parámetro.
<u>CDTray</u>	Abre o cierra la bandeja del CD.
<u>Ceiling</u>	Devuelve un número redondeado por encima del próximo entero.
<u>Chr</u>	Devuelve el carácter correspondiente a un código ASCII.
<u>ChrW</u>	Devuelve el carácter correspondiente a un código unicode.
<u>ClipGet</u>	Recupera texto del portapapeles.
<u>ClipPut</u>	Escribe texto en el portapapeles.

<u>ConsoleRead</u>	Lee datos desde la secuencia STDIN del proceso de script de Autolt.
<u>ConsoleWrite</u>	Escribe datos en la secuencia de STDOUT. Algunos editores de texto pueden leer dicha secuencia como otros programas.
<u>ConsoleWriteError</u>	Escribe datos en la secuencia de STDERR. Algunos editores de texto pueden leer dicha secuencia como otros programas.
<u>ControlClick</u>	Envía un clic de mouse a un determinado control.
<u>ControlCommand</u>	Envía un comando a un control.
<u>ControlDisable</u>	Deshabilita un control.
<u>ControlEnable</u>	Habilita un control.
<u>ControlFocus</u>	Establece el foco de entrada en un control dentro de una ventana.
<u>ControlGetFocus</u>	Devuelve el # de referencia del control que ha tomado el foco dentro de una ventana.
<u>ControlGetHandle</u>	Devuelve el apuntador interno (handle) de un control.
<u>ControlGetPos</u>	Devuelve el tamaño y posición de un control relativo a la ventana.
<u>ControlGetText</u>	Recupera el texto de un control.
<u>ControlHide</u>	Oculta un control.
<u>ControlListView</u>	Envía un comando a un control ListView32.
<u>ControlMove</u>	Mueve un control dentro de una ventana.
<u>ControlSend</u>	Envía una cadena de caracteres a un control.
<u>ControlSetText</u>	Establece texto de un control.
<u>ControlShow</u>	Muestra un control que ha sido ocultado.
<u>ControlTreeView</u>	Envía un comando a un control TreeView32.
<u>Cos</u>	Calcula el coseno de un número.
<u>Dec</u>	Devuelve la representación numérica de una cadena hexadecimal.

<u>DirCopy</u>	Copia un directorio conjuntamente con sus subdirectorios y archivos (Similar a xcopy).
<u>DirCreate</u>	Crea un directorio/carpeta.
<u>DirGetSize</u>	Devuelve el tamaño en bytes de un directorio dado.
<u>DirMove</u>	Mueve un directorio y todos sus subdirectorios y ficheros.
<u>DirRemove</u>	Borra un directorio/carpeta.
<u>DllCall</u>	Invocación dinámica de una función en una DLL.
<u>DllCallbackFree</u>	Libera un apuntador creado previamente con DllCallbackRegister.
<u>DllCallbackGetPtr</u>	Devuelve el puntero para una función callback que puede ser pasado al API Win32.
<u>DllCallbackRegister</u>	Crea una función DLL Callback definida por usuario.
<u>DllClose</u>	Cierra un DLL previamente abierto.
<u>DllOpen</u>	Abre un archivo DLL para usar en DllCall.
<u>DllStructCreate</u>	Crea una estructura en estilo C/C++ para ser usado en DllCall.
<u>DllStructGetData</u>	Devuelve el dato de un elemento de la estructura.
<u>DllStructGetPtr</u>	Devuelve el apuntador de la estructura o del elemento dentro de la estructura.
<u>DllStructGetSize</u>	Devuelve el tamaño de una estructura DLL en bytes.
<u>DllStructSetData</u>	Establece el dato en uno de los elementos en la estructura.
<u>DriveGetDrive</u>	Devuelve un arreglo devolviendo las unidades enumeradas.
<u>DriveGetFileSystem</u>	Devuelve el Tipo de Sistema de Archivo de una unidad.
<u>DriveGetLabel</u>	Devuelve la Etiqueta de Volumen de una unidad, si lo posee.
<u>DriveGetSerial</u>	Devuelve el Número de Serie de una unidad.
<u>DriveGetType</u>	Devuelve tipo de unidad.
<u>DriveMapAdd</u>	Mapea una unidad de red.
<u>DriveMapDel</u>	Desconecta una unidad de red.

<u>DriveMapGet</u>	Recupera información detallada de una unidad mapeada.
<u>DriveSetLabel</u>	Establece la Etiqueta de Volumen de una unidad.
<u>DriveSpaceFree</u>	Devuelve el espacio libre en disco(en Megabytes)de una ruta en Megabytes.
<u>DriveSpaceTotal</u>	Devuelve el espacio total libre en disco de una ruta en Megabytes.
<u>DriveStatus</u>	Devuelve el estado de una unidad como una cadena.
<u>EnvGet</u>	Recupera una variable de entorno.
<u>EnvSet</u>	Escribe en una variable de entorno.
<u>EnvUpdate</u>	Refresca el entorno del SO.
<u>Eval</u>	Devuelve el valor de una variable definida por una cadena.
<u>Execute</u>	Ejecuta una expresión.
<u>Exp</u>	Calcula la potencia de un número.
<u>FileChangeDir</u>	Cambia el directorio actual de trabajo.
<u>FileClose</u>	Cierra un fichero de texto previamente abierto.
<u>FileCopy</u>	Copia uno o más ficheros.
<u>FileCreateNTFSLink</u>	Crea un hardlink NTFS para un archivo o directorio.
<u>FileCreateShortcut</u>	Crea un acceso directo (.lnk) a un fichero.
<u>FileDelete</u>	Elimina uno o más ficheros.
<u>FileExists</u>	Chequea si un archivo o directorio existe.
<u>FileFindFirstFile</u>	Devuelve un identificador de búsqueda de acuerdo con una cadena de búsqueda.
<u>FileFindNextFile</u>	Devuelve un nombre de archivo de acuerdo con una invocación previa a FileFindFirstFile.
<u>FileFlush</u>	Limpia el buffer de una fichero en disco.
<u>FileGetAttrib</u>	Devuelve una cadena codificada conteniendo los atributos de un fichero.
<u>FileGetEncoding</u>	Determina la codificación de texto utilizada en un fichero.

<u>FileGetLongName</u>	Devuelve el nombre completo (dirección+nombre) de una ruta dada.
<u>FileGetPos</u>	Devuelve la posición actual en un fichero
<u>FileGetShortcut</u>	Devuelve detalles de un acceso directo.
<u>FileGetShortName</u>	Devuelve el nombre corto 8.3 de una dirección+nombre de archivo dado.
<u>FileGetSize</u>	Devuelve el tamaño de un archivo en bytes.
<u>FileGetTime</u>	Devuelve información del tiempo y la fecha para un fichero.
<u>FileGetVersion</u>	Devuelve información de la versión del "fichero".
<u>FileInstall</u>	Incluye e instala un archivo con el archivo compilado.
<u>FileMove</u>	Mueve uno o más directorios.
<u>FileOpen</u>	Abre un archivo de texto para lectura o escritura.
<u>FileOpenDialog</u>	Inicia un cuadro de diálogo de Abrir Archivo.
<u>FileRead</u>	Lee un número de caracteres desde un fichero previamente abierto.
<u>FileReadLine</u>	Lee una línea de texto desde un fichero previamente abierto.
<u>FileRecycle</u>	Envía un archivo o directorio a la papelera de reciclaje (recycle bin).
<u>FileRecycleEmpty</u>	Vacia la papelera de reciclaje.
<u>FileSaveDialog</u>	Inicializa un diálogo de Salvar Fichero.
<u>FileSelectFolder</u>	Inicializa un diálogo de selección de carpeta.
<u>FileSetAttrib</u>	Establece los atributos de uno o más ficheros.
<u>FileSetPos</u>	Establece la posición actual en el fichero.
<u>FileSetTime</u>	Establece la marca de tiempo de uno o más ficheros
<u>FileWrite</u>	Agrega un texto/dato al final de un fichero previamente abierto.
<u>FileWriteLine</u>	Agrega una línea de texto al final de un fichero de texto previamente abierto.

<u>Floor</u>	Devuelve un número por debajo del entero más próximo.
<u>FtpSetProxy</u>	Establece el proxy de internet para usar en acceso FTP.
<u>GUICreate</u>	Crea una ventana GUI.
<u>GUICtrlCreateAvi</u>	Crea un control "avi de video" en el GUI.
<u>GUICtrlCreateButton</u>	Crea un control Botón en el GUI.
<u>GUICtrlCreateCheckbox</u>	Crea un control de caja de chequeo (Checkbox) en el GUI.
<u>GUICtrlCreateCombo</u>	Crea un control ComboBox en el GUI.
<u>GUICtrlCreateContextMenu</u>	Crea un menú contextual para un control o una ventana GUI entera.
<u>GUICtrlCreateDate</u>	Crea un control de fecha en la GUI.
<u>GUICtrlCreateDummy</u>	Crea un control Dummy en la GUI.
<u>GUICtrlCreateEdit</u>	Crea un control Edit en la GUI.
<u>GUICtrlCreateGraphic</u>	Crea un control Graphic en la GUI.
<u>GUICtrlCreateGroup</u>	Crea un control de Grupo en la GUI.
<u>GUICtrlCreateIcon</u>	Crea un control Icono en la GUI.
<u>GUICtrlCreateInput</u>	Crea un control Input en la GUI.
<u>GUICtrlCreateLabel</u>	Crea un control de etiqueta estática en la GUI.
<u>GUICtrlCreateList</u>	Crea un control List en la GUI.
<u>GUICtrlCreateListView</u>	Crea un control ListView en la GUI.
<u>GUICtrlCreateListViewItem</u>	Crea un item en un control ListView.
<u>GUICtrlCreateMenu</u>	Crea un control Menu en la GUI.
<u>GUICtrlCreateMenuItem</u>	Crea un control MenuItem en la GUI.
<u>GUICtrlCreateMonthCal</u>	Crea un control de calendario para el GUI.
<u>GUICtrlCreateObj</u>	Crea un control ActiveX en la GUI.
<u>GUICtrlCreatePic</u>	Crea un control de Imagen para el GUI.
<u>GUICtrlCreateProgress</u>	Crea un control de barra de progreso en la GUI.

GUICtrlCreateRadio	Crea un control botón de Radio en la GUI.
GUICtrlCreateSlider	Crea un control Slider en la GUI.
GUICtrlCreateTab	Crea un control de Tab en la GUI.
GUICtrlCreateTabItem	Crea un control TabItem en la GUI.
GUICtrlCreateTreeView	Crea un control TreeView en la GUI.
GUICtrlCreateTreeViewItem	Crea un control TreeViewItem en la GUI.
GUICtrlCreateUpdown	Crea un control UpDown en la GUI.
GUICtrlDelete	Elimina un control.
GUICtrlGetHandle	Devuelve el identificador (handle) de un control y algún identificador especial (de item) en (Menu, ContextMenu, TreeViewItem).
GUICtrlGetState	Obtiene el estado actual de un control.
GUICtrlRead	Lee el estado o dato de un control.
GUICtrlRecvMsg	Envía un mensaje a un control y recoge información en IParam.
GUICtrlRegisterListViewSort	Registra una función definida de usuario (UDF) para ejecutarse durante un ordenamiento interno de un control listview.
GUICtrlSendMsg	Envía un mensaje a un control.
GUICtrlSendToDummy	Envía un mensaje a un control Dummy
GUICtrlSetBkColor	Establece el color de fondo de un control.
GUICtrlSetColor	Establece el color del texto de un control.
GUICtrlSetCursor	Establece el icono del cursor del mouse para un control en particular.
GUICtrlSetData	Modifica la información para un control.
GUICtrlSetDefBkColor	Establece el color de fondo por defecto en todos los controles de una ventana GUI.
GUICtrlSetDefColor	Establece el color de texto por defecto para todos los controles en una ventana GUI.
GUICtrlSetFont	Establece la fuente para un control.

<u>GUICtrlSetGraphic</u>	Modifica el dato para un control.
<u>GUICtrlSetImage</u>	Establece la imagen bitmap o icono para ser usado en un control.
<u>GUICtrlSetLimit</u>	Limita el número de caracteres/píxeles para un control.
<u>GUICtrlSetOnEvent</u>	Define una Función de usuario (UDF) para ser invocada cuando damos clic sobre un control.
<u>GUICtrlSetPos</u>	Cambia la posición de un control dentro de una ventana GUI.
<u>GUICtrlSetResizing</u>	Define el método de reajuste usado por un control.
<u>GUICtrlSetState</u>	Cambia el estado de un control.
<u>GUICtrlSetStyle</u>	Cambia el estilo de un control.
<u>GUICtrlSetTip</u>	Establece el texto de tip asociado a un control.
<u>GUIDelete</u>	Borra una ventana GUI con todos los controles que esta contiene.
<u>GUIGetCursorInfo</u>	Obtiene la posición del cursor relativo a una ventana GUI.
<u>GUIGetMsg</u>	Sondea la GUI para detectar la ocurrencia de algún evento.
<u>GUIGetStyle</u>	Devuelve el estilo de una ventana GUI.
<u>GUICreateMsg</u>	Registra una función de usuario (UDF) para un ID de mensaje (WM_MSG) de ventana conocido.
<u>GUICreateAccelerators</u>	Establece la tabla de opciones aceleradas de una ventana GUI.
<u>GUICreateBkColor</u>	Establece el color de fondo de una ventana GUI.
<u>GUICreateCoord</u>	Establece coordenadas absolutas para el Control siguiente.
<u>GUICreateCursor</u>	Establece el ícono del puntero para una ventana GUI.
<u>GUICreateFont</u>	Establece la fuente por defecto para una ventana GUI.
<u>GUICreateHelp</u>	Establece el archivo ejecutable que será ejecutado cuando se opriime la tecla F1.
<u>GUICreateIcon</u>	Establece el ícono usado en la ventana GUI.
<u>GUICreateOnEvent</u>	Define un función de usuario (UDF) para ser llamada cuando sobre un botón del Sistema damos clic.

<u>GUISetState</u>	Cambia el estado de una ventana GUI
<u>GUISetStyle</u>	Cambia el estilos de una ventana GUI.
<u>GUIStartGroup</u>	Define el agrupamiento de algunos controles subsecuentemente creados.
<u>GUISwitch</u>	Altera la actual ventana por otra ventana GUI.
<u>Hex</u>	Devuelve la representación de una cadena o valor binario convertido a hexadecimal.
<u>HotKeySet</u>	Establece el método de teclado para una función de usuario determinada.
<u>HttpSetProxy</u>	Establece el proxy de internet para tener acceso vía http.
<u>HttpSetUserAgent</u>	Establece la cadena user-agent enviada con peticiones InetGet() y InetRead().
<u>HWnd</u>	Convierte una expresión en un apuntador (handle) HWND.
<u>InetClose</u>	Cierra un identificador (handle) devuelto desde InetGet().
<u>InetGet</u>	Descarga una archivo de Internet usando los protocolos HTTP, HTTPS o FTP
<u>InetGetInfo</u>	Devuelve detalles de datos para un identificador (handle) devuelto desde InetGet().
<u>InetGetSize</u>	Devuelve el tamaño (en bytes) de un archivo localizado en internet.
<u>InetRead</u>	Descarga un fichero desde internet usando los protocolos HTTP, HTTPS o FTP.
<u>IniDelete</u>	Borra un valor de un fichero .ini con formato estándar.
<u>IniRead</u>	Lee un valor desde un fichero .ini con formato estándar.
<u>IniReadSection</u>	Lee todos los pares de valores/claves de una sección en un fichero .ini con formato estándar.
<u>IniReadSectionNames</u>	Lee todas las secciones en un fichero .ini con formato estándar.
<u>IniRenameSection</u>	Renombra una sección en un fichero .ini con formato estándar.
<u>IniWrite</u>	Escribe un valor para un fichero .ini con formato estándar.

<u>IniWriteSection</u>	Escribe una sección para un fichero .ini con formato estándar.
<u>InputBox</u>	Muestra una caja de entrada que requiere del usuario la entrada de una cadena.
<u>Int</u>	Devuelve la representación numérica del valor entero (el número completo) de una expresión.
<u>IsAdmin</u>	Verifica que el usuario actual tenga privilegio de administrador.
<u>IsArray</u>	Chequea si una variable es de tipo arreglo.
<u>IsBinary</u>	Chequea si una variable o expresión es de tipo binaria.
<u>IsBool</u>	Chequea si una variable es de tipo booleano.
<u>IsDeclared</u>	Chequea si una variable ha sido declarada.
<u>IsDIIStruct</u>	Chequea si una variable es de tipo DIIStruct.
<u>IsFloat</u>	Chequea si una variable o expresión es de tipo flotante (decimal).
<u>IsHWnd</u>	Chequea si una variable es un puntero y un apuntador válido de ventana.
<u>IsInt</u>	Chequea si una variable o expresión es de tipo entero.
<u>IsKeyword</u>	Chequea si una variable es una palabra clave (Por ejemplo, Default).
<u>IsNumber</u>	Chequea si una variable es de tipo numérico.
<u>IsObj</u>	Chequea si una variable o expresión es de tipo objeto.
<u>IsPtr</u>	Chequea si una variable es de tipo puntero.
<u>IsString</u>	Chequea si una variable es de tipo cadena.
<u>Log</u>	Calcula el logaritmo natural de un número.
<u>MemGetStats</u>	Devuelve información relativa a la memoria.
<u>Mod</u>	Ejecuta una operación de módulo.
<u>MouseClick</u>	Realiza una operación de clic del mouse.
<u>MouseClickDrag</u>	Realiza una operación clic-y-arrastre con el mouse.
<u>MouseDown</u>	Realiza un evento de presionar el mouse hacia abajo en la

	posición actual del mouse.
<u>MouseGetCursor</u>	Devuelve el número de ID del cursor actual del mouse.
<u>MouseGetPos</u>	Recupera la posición actual de mouse.
<u>MouseMove</u>	Mueve el cursor del mouse.
<u>MouseUp</u>	Realiza un evento de presionar el mouse hacia arriba en la posición actual del mouse.
<u>MouseWheel</u>	Mueve la rueda del mouse arriba o abajo. Solamente en NT/2000/XP.
<u>MsgBox</u>	Muestra un simple cuadro de mensaje con tiempo opcional para expirar.
<u>Number</u>	Devuelve la representación numérica de una expresión.
<u>ObjCreate</u>	Crea una referencia a un objeto COM del nombre de clase dado.
<u>ObjEvent</u>	Maneja eventos entrantes del Objeto dado.
<u>ObjGet</u>	Recupera una referencia a un objeto COM de un proceso existente o nombre de archivo.
<u>ObjName</u>	Devuelve el nombre o descripción de interfaz de un Objeto.
<u>OnAutoItExitRegister</u>	Registra una función para ser llamada cuando AutoIt inicia.
<u>OnAutoItExitUnRegister</u>	Des-registra una función que es llamada cuando AutoIt finaliza.
<u>Ping</u>	Envía pings a un host y devuelve el tiempo de ida-y-vuelta.
<u>PixelChecksum</u>	Genera una suma de verificación(checksum) para una región de pixeles.
<u>PixelGetColor</u>	Devuelve el color de un pixel de acuerdo a las coordenadas x,y del pixel.
<u>PixelSearch</u>	Busca un rectángulo de pixeles el color de pixel especificado.
<u>PluginClose</u>	Cerrar un fichero plugin.
<u>PluginOpen</u>	Abre un fichero de plugin.
<u>ProcessClose</u>	Termina un proceso nombrado.

<u>ProcessExists</u>	Revisa si un proceso especificado existe.
<u>ProcessGetStats</u>	Devuelve un arreglo sobre Memoria o información de IO(Entrada y Salida) de un proceso en ejecución.
<u>ProcessList</u>	Devuelve un arreglo listando los procesos actualmente en ejecución (nombres y PIDs)
<u>ProcessSetPriority</u>	Cambia la prioridad de un proceso
<u>ProcessWait</u>	Pausa el script ejecución hasta que proceso dado exista.
<u>ProcessWaitClose</u>	Pausa el script en ejecución hasta que dado proceso termine.
<u>ProgressOff</u>	Finaliza la ventana de Progreso.
<u>ProgressOn</u>	Crea una ventana personalizable de barra de progreso.
<u>ProgressSet</u>	Establece la posición y/o texto de una ventana de Progreso creada anteriormente.
<u>Ptr</u>	Convierte una expresión en una variante de puntero.
<u>Random</u>	Genera un número seudo-aleatorio del tipo flotante.
<u>RegDelete</u>	Elimina una clave del registro.
<u>RegEnumKey</u>	Lee el nombre de una subclave de acuerdo a su instancia.
<u>RegEnumVal</u>	Lee el nombre de un valor de acuerdo a su instancia.
<u>RegRead</u>	Lee un valor del registro.
<u>RegWrite</u>	Crea una clave o valor en el registro.
<u>Round</u>	Devuelve un número redondeado a un número determinado de cifras decimales.
<u>Run</u>	Ejecuta un programa externo.
<u>RunAs</u>	Ejecuta un programa externo bajo el contexto de un usuario diferente.
<u>RunAsWait</u>	Ejecuta un programa externo bajo el contexto de un usuario diferente y pausa la ejecución del script hasta que el programa finaliza.
<u>RunWait</u>	Ejecuta un programa externo y pausa la ejecución del script hasta que el programa finaliza.

<u>Send</u>	Envía pulsaciones simuladas de teclas a la ventana activa.
<u>SendKeepActive</u>	Intenta mantener activa una ventana específica durante Send().
<u>SetError</u>	Establece manualmente el valor de la macro @error.
<u>SetExtended</u>	Manualmente establece el valor del macro @extended.
<u>ShellExecute</u>	Ejecuta un programa externo usando la API de ShellExecute.
<u>ShellExecuteWait</u>	Ejecuta un programa externo usando la API de ShellExecute y pausa la ejecución del script hasta que finalice.
<u>Shutdown</u>	Apaga el sistema.
<u>Sin</u>	Calcula el seno de un número.
<u>Sleep</u>	Pausa la ejecución del script.
<u>SoundPlay</u>	Ejecuta un archivo de sonido.
<u>SoundSetWaveVolume</u>	Establece el volumen de onda del sistema por porcentaje.
<u>SplashImageOn</u>	Crea una ventana con imagen personalizada.
<u>SplashOff</u>	Desactiva una ventana de SplashText o SplashImage.
<u>SplashTextOn</u>	Crea una ventana personalizada de texto.
<u>Sqrt</u>	Calcula la raíz cuadrada de un número.
<u>SRandom</u>	Establece un semilla para la generación de números aleatorios.
<u>StatusbarGetText</u>	Recupera el texto de una barra de estado estándar.
<u>StderrRead</u>	Lee del flujo de STDERR de un proceso hijo previamente ejecutado.
<u>StdinWrite</u>	Escribe un número de caracteres al flujo de STDIN de un proceso hijo previamente ejecutado.
<u>StdioClose</u>	Cierra todos los recursos asociados con un proceso previamente ejecutado redirección de STDIO.
<u>StdoutRead</u>	Lee del flujo STDOUT de un proceso hijo previamente ejecutado.
<u>String</u>	Devuelve una representación de una expresión en cadena de

	texto.
<u>StringAddCR</u>	Toma una cadena de texto y todos los caracteres de línea prefijos (Chr(10)) con un carácter de retorno de carro (Chr(13))
<u>StringCompare</u>	Compara dos cadenas de texto con opciones.
<u>StringFormat</u>	Devuelve una cadena de texto con formato (similar a la función sprintf() de C)
<u>StringFromASCIIArray</u>	Convierte un arreglo de códigos ASCII a una cadena.
<u>StringInStr</u>	Revisa si una cadena contiene una subcadena dada.
<u>StringIsAINum</u>	Revisa si una cadena contiene solamente caracteres alfanuméricos.
<u>StringIsAlpha</u>	Revisa si una cadena contiene solamente caracteres alfabéticos.
<u>StringIsASCII</u>	Revisa si una cadena contiene solamente caracteres ASCII en el rango 0x00 - 0x7f (0 - 127).
<u>StringIsDigit</u>	Revisa si una cadena contiene solamente caracteres de dígitos (0-9).
<u>StringIsFloat</u>	Revisa si una cadena es un número tipo punto flotante.
<u>StringIsInt</u>	Revisa si una cadena es un entero.
<u>StringIsLower</u>	Revisa si una cadena contiene solamente letras minúsculas.
<u>StringIsSpace</u>	Revisa si una cadena contiene solamente caracteres de espacios en blanco.
<u>StringIsUpper</u>	Revisa si una cadena contiene solamente letras mayúsculas.
<u>StringIsXDigit</u>	Revisa si una cadena contiene solamente dígitos (0-9) y caracteres hexadecimales (A-F)
<u>StringLeft</u>	Devuelve un número de caracteres a partir del lado izquierdo de una cadena de texto.
<u>StringLen</u>	Devuelve el número de caracteres en una cadena de texto.
<u>StringLower</u>	Convierte a cadena a minúsculas.
<u>StringMid</u>	Extrae un número de caracteres de una cadena de texto.

<u>StringRegExp</u>	Revisa si una cadena encaja en un patrón dado de expresión regular.
<u>StringRegExpReplace</u>	Reemplaza texto en un cadena basado en un expresión regular.
<u>StringReplace</u>	Reemplaza subcadenas en una cadena.
<u>StringRight</u>	Devuelve un número de caracteres del lado izquierdo de una cadena.
<u>StringSplit</u>	Divide una cadena en varias partes de acuerdo a un delimitador dado.
<u>StringStripCR</u>	Remueve todos los valor de retorno de carro (Chr(13)) de una cadena.
<u>StringStripWS</u>	Recorta los espacios en blanco de una cadena.
<u>StringToASCIIArray</u>	Convierte una cadena a un arreglo contenido el código ASCII de cada carácter.
<u>StringToBinary</u>	Convierte una cadena en datos binarios.
<u>StringTrimLeft</u>	Corta un número de caracteres del lado izquierdo de una cadena.
<u>StringTrimRight</u>	Corta un número de caracteres del lado derecho de una cadena.
<u>StringUpper</u>	Convierte una cadena a mayúsculas.
<u>Tan</u>	Calcula la tangente de un número.
<u>TCPSAccept</u>	Permite una conexión entrante intentando en un socket.
<u>TCPCloseSocket</u>	Cierra un socket de TCP.
<u>TCPConnect</u>	Crea un socket conectado un servidor existente.
<u>TCPListen</u>	Crea un socket en espera para una conexión entrante.
<u>TCPNameToIP</u>	Convierte un nombre de Internet a dirección de IP.
<u>TCPRecv</u>	Recibe datos de un socket conectado.
<u>TCPSend</u>	Envía datos en un socket conectado.
<u>TCPShutdown,</u> <u>UDPSshutdown</u>	Detiene los servicios de TCP/UDP.

<u>TCPStartup, UDPStartup</u>	Empieza los servicios de TCP o UDP.
<u>TimerDiff</u>	Devuelve la diferencia en tiempo de una llamada previa a TimerInit().
<u>TimerInit</u>	Devuelve una marcaDeTiempo (en milisegundos)
<u>ToolTip</u>	Crea un tooltip en cualquier parte de la pantalla.
<u>TrayCreateItem</u>	Crea un control de itemMenu para la bandeja.
<u>TrayCreateMenu</u>	Crea un control de menu para el menu de bandeja.
<u>TrayGetMsg</u>	Consulta cualquier evento ocurrido en un menú de bandeja.
<u>TrayItemDelete</u>	Elimina un control de menu/item del menú de bandeja.
<u>TrayItemGetHandle</u>	Devuelve el apuntador para un menú de bandeja(item).
<u>TrayItemGetState</u>	Obtiene el estado actual de un control.
<u>TrayItemGetText</u>	Obtiene el itemtexto de un control de menú/item de bandeja.
<u>TrayItemSetOnEvent</u>	Define un Función de usuario para ser invocada cuando un item del ícono de Autoit es clickeado.
<u>TrayItemSetState</u>	Establece el estado de un control menu/item en la bandeja del sistema.
<u>TrayItemSetText</u>	Establece el texto del item en un control menu/item.
<u>TraySetClick</u>	Establece el modo de clic en el ícono de la bandeja - con clics del mouse se mostrará el menú del icono.
<u>TraySetIcon</u>	Carga/establece un icono especificado en la bandeja.
<u>TraySetOnEvent</u>	Define una función de usuario que será invocada cuando una acción especial ocurre en la bandeja.
<u>TraySetPauselcon</u>	Carga/establece un icono de pausa en la bandeja especificado.
<u>TraySetState</u>	Establece el estado del icono de la bandeja del sistema.
<u>TraySetToolTip</u>	(Re)Establece el texto tooltip para el icono de bandeja.
<u>TrayTip</u>	Muestra un globo tip desde el icono de Autolt. (2000/XP solamente)
<u>UBound</u>	Devuelve el tamaño de un arreglo dimensionado.

<u>UDPBind</u>	Crea un socket delimitado para una conexión entrante.
<u>UDPCloseSocket</u>	Cierra un socket UDP.
<u>UDPOpen</u>	Abre un socket conectado para un servidor existente.
<u>UDPRecv</u>	Recibe datos de un socket abierto.
<u>UDPSend</u>	Envía un dato por un socket abierto.
<u>VarGetType</u>	Devuelve el tipo de representación interna de un dato variante.
<u>WinActivate</u>	Activa (toma el foco) una ventana.
<u>WinActive</u>	Chequea si una ventana está abierta y ha sido actualmente activada.
<u>WinClose</u>	Cierra una ventana
<u>WinExists</u>	Chequea si una ventana especificada existe.
<u>WinFlash</u>	Provoca parpadeo de una ventana en la barra de tareas.
<u>WinGetCaretPos</u>	Devuelve las coordenadas de intercalamiento para la ventana en primer plano.
<u>WinGetClassList</u>	Captura las clases internas de una ventana.
<u>WinGetClientSize</u>	Captura el tamaño de una ventana determinada de área cliente.
<u>WinGetHandle</u>	Devuelve el identificador interno de una ventana.
<u>WinGetPos</u>	Devuelve la posición y tamaño de una ventana dada.
<u>WinGetProcess</u>	Devuelve el ID del proceso (PID) asociado a una ventana.
<u>WinGetState</u>	Devuelve el estado de una ventana dada.
<u>WinGetText</u>	Devuelve el texto de una ventana.
<u>WinGetTitle</u>	Devuelve el título completo de la ventana.
<u>WinKill</u>	Fuerza una ventana a cerrar.
<u>WinList</u>	Devuelve una lista de ventanas.
<u>WinMenuItemSelect</u>	Invoca el ítem de un menú dentro de una ventana.

<u>WinMinimizeAll</u>	Minimiza todas las ventanas.
<u>WinMinimizeAllUndo</u>	Deshace la operación de la función WinMinimizeAll.
<u>WinMove</u>	Mueve y/o cambia el tamaño de una ventana.
<u>WinSetOnTop</u>	Cambia una ventana colocándole el atributo "Siempre Visible" (Always On Top)
<u>WinSetState</u>	Muestra, Oculta, minimiza, maximiza, o restaura una ventana.
<u>WinSetTitle</u>	Cambia el título de una ventana.
<u>WinSetTrans</u>	Establece el nivel de trasparencia de una ventana. (Windows 2000/XP o superior)
<u>WinWait</u>	Pausa la ejecución del script hasta que la ventana solicitada exista.
<u>WinWaitActive</u>	Pausa la ejecución de un script hasta que la ventana solicitada se activa.
<u>WinWaitClose</u>	Pausa la ejecución del script hasta que la ventana solicitada no exista.
<u>WinWaitNotActive</u>	Pausa la ejecución de un scripts hasta que la ventana especificada no esté activa.

Funciones del Usuario

Las funciones del usuario son declaradas usando [Func...EndFunc](#).

Pueden aceptar parámetros y retornar valores según lo requieran.

Los nombres deben comenzar con una letra cualquiera, o bien con un guión bajo y pueden contener cualquier combinación de letras (sin acentos!), números y guiones. Algunos nombres válidos para funciones:

```
MiFunc
Func1
_Mi_Funcion
```

Un ejemplo de uso para una función que duplica un número por 10 veces:

```
$val = 10
For $i = 1 To 10
  $doble = MiDoble($val)
```

```
MsgBox(0, "", $val & " el doble es " & $doble)  
$val = $doble
```

Next

Exit

;función que duplica un número cualquiera

```
Func MiDoble($valor)
```

```
$valor = $valor * 2
```

```
Return $valor
```

EndFunc

Notas de Funciones

Muchas funciones contienen parámetros opcionales que pueden ser omitidos. Si desea especificar un parámetro opcional, todos los parámetros anteriores deben ser especificados! Por ejemplo, considere [Run](#) ("nombredearchivo", ["dirDeTrabajo" [, bandera]]). Si se desea especificar la *bandera*, se **debe** especificar *directorioDeTrabajo* .

Muchas funciones Win___ contienen un parámetro opcional llamado "texto". Este parámetro está destinado para ayudar a diferenciar entre ventanas que tienen títulos idénticos.

Algunas funciones indican éxito/fallo como **valor devuelto**; otros se indican fijando en la **flag de @error**. Ambos tienen las siguientes significación... @error = 0 ; cuando es igual a cero, es siempre exitoso Return = varía, pero típicamente valores no-cero exitosos permiten leer fácilmente el código...

```
If algunaFuncUsuario() then ; ...función trabajó  
If Not someUserFunc() then ; ...función falló  
$x = FileReadLine("C:\someFile.txt")  
If @error = -1 Then ;fin-de-archivo fue lanzado
```

Si una función puede establecer el flag de @error, se debería revisar siempre antes de usar un valor devuelto - si @error indica un error entonces el valor devuelto de la función es generalmente indefinido...

@error es siempre establecido a 0 cuando entra en una función.

Cuando la documentación establece que el valor devuelto es = ninguno, AutoIt siempre devuelve un valor para evitar errores. Valor de 1 es usualmente devuelto, pero no debería depender de este valor devuelto.

Cuando un parámetro opcional necesita ser definido y es precedido por uno o más parámetros opcionales, los parámetros anteriores deben ser dados. Esto tal vez sea "" para un parámetro de cadena de texto y -1 para otros tipos. Algunas funciones como StringInStr o StringReplace requieren 0. Ver la descripción correspondiente del parámetro opcional en cada caso.

Referencia del Lenguaje - Comentarios

Aunque se permite solamente una declaración por línea, una que fuera extensa puede ser separada en líneas múltiples mediante un guión bajo " _" precedido por un espacio en blanco al final del "quiebre" de la línea. Una definición de cadena no puede ser separada en diferentes líneas, a menos que se utilice una concatenación:

```
MsgBox(4096,"","Soy una línea de texto muy larga, así que" &_
"simplemente me dividieron en dos partes utilizando este recurso.")
```

El punto y coma (;) es el carácter utilizado para efectuar comentarios. A menos que esté en el interior de una cadena, todo texto que sea escrito a continuación en la misma línea, será ignorado por el intérprete/compilador.

```
; La próxima línea contiene un comentario explicativo al final
Sleep(5000) ;pausa 5 segundos
```

Puede combinar el guión bajo con el punto y coma para introducir comentarios en líneas y aún tener una declaración extensa separada con respecto a la línea siguiente.

```
dim $b_ ; Este guión no es un carácter de continuación
dim $k_
Dim $a[8][2] = [_
[ "Palabra", 4 ], _ ; Comentario 1
[ "Prueba", 3 ], _
[ "pi", 3.14159], _ ; Asocia el nombre con el valor
[ "e", 2.718281828465], _ ; Aquí es lo mismo
[ "prueba;1;2;3", 123], _
[ ';, Asc(';') ], _ ; Este comentario es ignorado, pero las cadenas permanecen.
[ "", 0] ]
```

Es posible también introducir grandes bloques de comentario en el código usando las directivas [#comments-start](#) y [#comments-end](#).

#comments-start (#cs) #comments-end (#ce)

Especifica que una sección entera en el script debe ser tomado como comentario.

#comments-start

...

...

#comments-end

Parámetros

Ninguno

Comentarios

Las directivas `#comments-start` y `#comments-end` puede ser anidadas. Usted puede también usar las palabras claves abreviadas `#cs` y `#ce`.

Adicionalmente, pueden ser comentadas!

Ejemplo

`#comments-start`

`MsgBox(4096, "", "Esto no será ejecutado")`

`MsgBox(4096, "", "o esto")`

`#comments-end`

`///`

`#cs`

`MsgBox(4096, "", "Esto se imprime si '#cs' es quitado.")`

`#ce`

Referencia GUI

AutoIt posee la habilidad de crear simples Interfaces Gráficas de Usuario (GUIs) que consisten en una o más ventanas y controles.

Conceptos GUI

Una GUI consiste en una o más ventanas y cada ventana contiene uno o más controles. Una GUI es un "manejador de eventos" capaz de reaccionar ante ciertos eventos que se producen sobre ella - como cuando un botón es clickeado. La GUI se encuentra en un estado de espera hasta que un evento es desencadenado. Algo semejante a estar en la puerta esperando al cartero - te sientes esperar una carta y luego decides que hacer con ella - así trabaja exactamente una GUI - usted espera venir al cartero.

Por supuesto, usted puede elegir hacer otras tareas mientras la GUI permanece activa - por ejemplo puede que utilice las funciones GUI para crear una barra de progreso que se actualiza mientras el script desarrolla complejas operaciones.

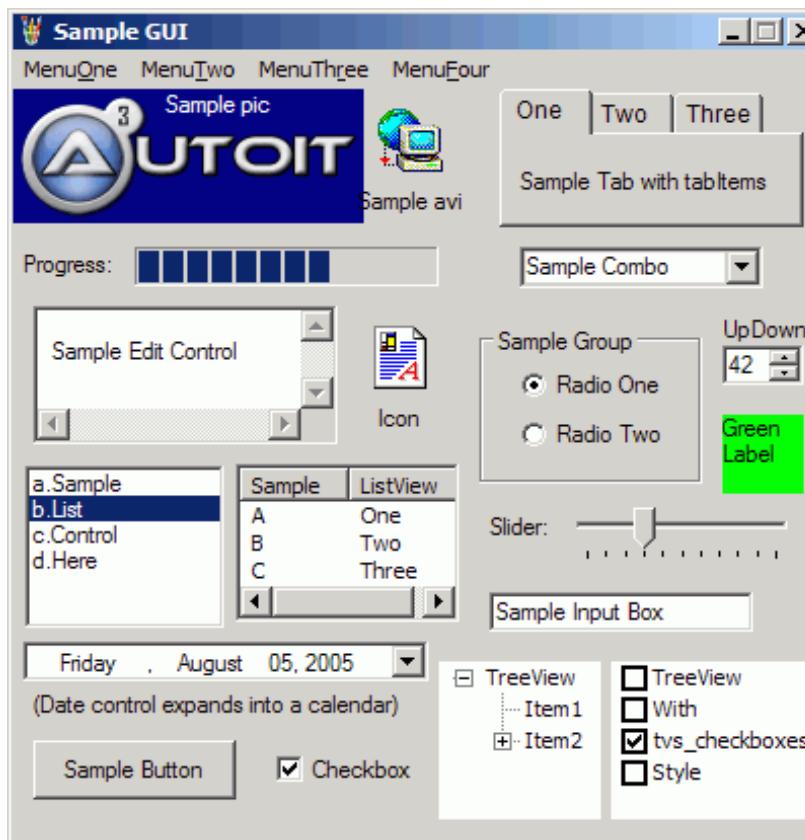
Controles GUI

Todos los usuarios deben conocer bien los controles - Un control es todo aquello dentro de una ventana que puede recibir un click o interaccionar con el usuario. Los tipos de controles que pueden ser creados con AutoIt se listan más abajo - La mayoría de ellos pueden ser encontrados en muchas ventanas de Windows.

Label	Una cadena de texto plano.
Button	Un Botón simple.
Input	Un simple control de línea que permite introducir texto dentro del mismo.
Edit	Un control multi-línea que permite introducir texto dentro del mismo.
Checkbox	Una caja de chequeo que puede ser "marcada" o "desmarcada".
Radio	Un botón circular de chequeo.
Combo	Una lista contenida en una caja.
List	Una lista.
Date	Un control calendario.
Pic	Una imagen.
Icon	Un ícono.
Progress	Una Barra de Progreso.

Tab	Un grupo de controles contenidos en pestañas.
UpDown	Un control que puede ser adicionado para entrada de controles.
Avi	Muestra un Clip en formato AVI.
Menu	Un menú en la parte superior de la ventana.
ContextMenu	Un menú que aparece cuando damos click dentro de la ventana.
TreeView	Un control similar al control Explorador de Windows.
Slider	Un control similar al control del volumen de Windows.
ListView	Un control que muestra información en columnas.
ListViewItem	Un control que muestra ítems en un control listView.
Graphic	Un control que muestra gráficos dibujados con GUICtrlSetGraphic.
Dummy	Un control de usuario dummy.

Aquí se muestra una simple ventana GUI que contiene algunos de los controles disponibles. Como usted nota es posible crear una extensa variedad de GUIs detallados!



Los controles son creados con la función [GUICtrlCreate...](#). Cuando un control es creado un **ID de Control** es devuelto. Lo más importante a tener en cuenta con el ID de Control es:

El ID de control es un número positivo (esto es, un número mayor que 0)

Cada ID de control es único - aun cuando existan múltiples ventanas.

El ID de control es realmente el valor del Id de control que se muestra en la 'herramienta de información de ventana de Autoit' [AutoIt Window Info Tool](#) ..

Funciones GUI Básicas

Estas son las principales funciones que usted necesita para crear una GUI. Esto es lo básico, si eres listo existen muchas más que le permitirán realizar avanzados GUIs.

Function	Explanation
GUICreate	Crea una ventana.
GUICtrlCreate...	Crea varios controles en la ventana.
GUISetState	Muestra u oculta la ventana.
GUIGetMsg	Sondea la GUI para ver si un evento ha ocurrido (Solo modo MessageLoop)
GUICtrlRead	Lee el dato de un control.
GUICtrlSetData	Establece/cambia el dato en un control.
href=..../functions/GUICtrlUpdateManagement.htm">GUICtrlUpdate...	Cambia varias opciones para un control (color, estilo, etc.)

Usted necesita #include <GUIConstantsEx.au3> para constantes relativas a GUI. Existen [otros ficheros includes](#) que contienen constantes relativas a varios controles que usted puede usar en su GUI.

Antes de escribir cualquier scripts GUI usted debe incluir el fichero **GUIConstants.au3** - este contiene todas las variables y constantes necesarias.

Primero crearemos una ventana, llamada "Hola Mundo" esta tendrá 200 por 100 pixeles de dimensión. Cuando se crea una ventana nueva esta se oculta - debemos establecer el atributo "Mostrar" (show) para que la ventana pueda ser visible.

```
#include <GUIConstants.au3>
GUICreate("Hola Mundo", 200, 100)
GUISetState(@SW_SHOW)
Sleep(2000)
```

El ejemplo anterior mostrará la ventana por dos segundos. Esto no resulta interesante...ahora vamos a adicionar un texto y un botón OK. El texto será adiconado en la posición 30, 10 y el botón en 70, 50 con un ancho de 60 pixeles.

```
include <GUIConstants.au3>
GUICreate("Hello World", 200, 100)
GUICtrlCreateLabel("Hola Mundo! Como estás?", 30, 10)
GUICtrlCreateButton("OK", 70, 50, 60)
GUISetState(@SW_SHOW)
```

Sleep(2000)

Esto es muy bueno!, ¿Como hacemos ahora para que reaccione cuando hagamos un clik sobre el botón? Bueno, este es el momento en que debemos decidir por cual vía se van a procesar los eventos - vía lazo de mensajes (**MessageLoop**) o por la funciones **OnEvent**.

Modos de Eventos GUI

Como mencionamos más arriba estos son los dos modos GUI básicos: modo **Lazo de mensajes (MessageLoop)** y modo **OnEvent**. Los modos son simplemente dos maneras diferentes de reaccionar a los eventos de la ventana. El modo que usted escoja dependerá de sus *preferencias personales*, y hasta cierto punto también del tipo de GUI que usted vaya a crear. Ambos modos poseen igual capacidad en la creación de cualquier GUI pero en ocasiones un modo resulta más conveniente para una operación determinada.

El modo por defecto es el modo de lazo de mensajes. Para cambiar al modo OnEvent defina [Opt\("GUIONEventMode", 1\)](#) al inicio del script.

Modo de lazo-de mensajes (modo por defecto)

En el modo lazo de mensajes (Message-loop) su script se halla la mayoría del tiempo en un ciclo constante. Este ciclo sondeará la GUI usando la función [GUIGetMsg](#). Cuando un evento ocurre se devuelve un valor por la función GUIGetMsg que muestra el detalle del evento (un botón es clickeado, La GUI ha sido cerrada (close), etc.)

En este modo usted solo recibe el evento mientras este activo el chequeo por la función GUIGetMsg debe asegurar que dicho chequeo se realice con una frecuencia alta pues de otro modo la GUI sería insensible.

Este modo es el mejor para la GUI donde la GUI es el "Rey" y todos los demás objetos están a su alrededor esperando un evento.

Ver [esta página](#) para una explicación más detallada de modo [MessageLoop](#)

Modo OnEvent

En el modo OnEvent existe un chequeo constante de la GUI hasta encontrar cualquier evento, entonces la GUI temporalmente se detiene y llama a una función definida por usuario (UDF) para dar respuesta al evento. Por ejemplo, si el usuario da click en el Botón1 de la GUI se detiene el chequeo y se llama a una función definida por usuario previa que atienda al Botón1. Cuando la función es ejecutada el script se resume nuevamente. Este modo es similar al método usado por forma en Visual Basic.

Este modo es el mejor para GUIs donde la GUI es de importancia secundaria y el script principal puede realizar otras tareas además de ocuparse de la GUI.

Ver [esta página](#) para una explicación más detallada del modo [OnEvent](#)

Observación: La diferencia que existe entre ambos modos de manejar eventos es la siguiente: En el modo Lazo de Mensajes la GUI **no se detiene** al ocurrir un evento, **esta sigue sondeando** a la espera de más eventos. En el modo OnEvent la GUI **se detiene** al ocurrir un evento y ejecuta una UDF, esto significa que la GUI **se hace temporalmente insensible** hasta

que no **finalice la UDF**, una vez finalizada, la GUI continúa con el sondeo a la espera de otro evento.

Referencia GUI - Modo lazo de mensajes (MessageLoop)

En el modo lazo de mensajes (Message-loop) el script se halla la mayoría del tiempo en un ciclo constante. Este ciclo sondeará la GUI usando la función [GUIGetMsg](#). Cuando un evento ocurre se devuelve un valor por la función GUIGetMsg que muestra el detalle del evento (un botón es clickeado, La GUI ha sido cerrada (close), etc.)

El modo Lazo de mensajes (MessageLoop) es el modo de mensajes por defecto de GUIs en AutoIt - el otro modo posible es el modo [OnEvent](#).

En este modo usted solo recibe el evento mientras este activo el chequeo por la función GUIGetMsg, debe asegurar que dicho chequeo se realice con una frecuencia alta pues de otro modo la GUI sería insensible.

Formato básico de Lazo de Mensajes

La forma general del código para Lazo de Mensajes es:

While 1

\$msg = GUIGetMsg()

...

...

WEnd

Usualmente un lazo de este tipo pondría la CUP al 100% - afortunadamente la función [GUIGetMsg](#) automáticamente libera la CPU cuando no hay eventos en espera. **No** ponga una 'espera' (Sleep) dentro del lazo pues esto podría saturar la CPU - esto solo causa que la GUI se haga insensible.

GUI Events

Existen tres tipos de mensajes de eventos que [GUIGetMsg](#) puede retornar:

Sin Evento

Eventos de Controles

Eventos del Sistema

Sin Evento

Cuando no hay eventos en espera de ser procesados, la función GUIGetMsg retorna **0**. En una GUI usual es *el evento más común*.

Eventos de Controles

Cuando un control es Clickeado o cambia se envía el evento - este evento es un número **positivo que corresponde al IDControl** que es devuelto por la función [GUICtrlCreate....](#) cuando es creado.

Eventos del Sistema

Son los Eventos del Sistema - Como cuando es cerrada la GUI - son números **negativos**. Los diferentes eventos de este tipo son listados más abajo y se encuentran en [GUIConstants.au3](#):

```
$GUI_EVENT_CLOSE      ;La GUI ha sido cerrada  
$GUI_EVENT_MINIMIZE   ;La GUI ha sido minimizada  
$GUI_EVENT_RESTORE    ; La GUI ha sido restaurada  
$GUI_EVENT_MAXIMIZE   ; La GUI ha sido maximizada  
$GUI_EVENT_PRIMARYDOWN ;Se ha oprimido el botón primario del mouse  
$GUI_EVENT_PRIMARYUP    ;Se ha liberado el botón primario del mouse  
$GUI_EVENT_SECONDARYDOWN ;Se ha oprimido el botón secundario del mouse  
$GUI_EVENT_SECONDARYUP   ;Se ha liberado el botón secundario del mouse  
$GUI_EVENT_MOUSEMOVE     ; Se ha movido el mouse  
$GUI_EVENT_RESIZED       ; Ha cambiado de tamaño la GUI  
$GUI_EVENT_DROPPED        ; Se ha soltado un objeto
```

Ejemplos de GUI

En la página principal de la [Referencia GUI](#) iniciamos un simple ejemplo de "Hola Mundo" que puede verse aquí:

```
#include <GUIConstants.au3>  
GUICreate("Hola Mundo", 200, 100)  
GUICtrlCreateLabel("Hola Mundo! Cómo estás?", 30, 10)  
GUICtrlCreateButton("OK", 70, 50, 60)  
GUISetState(@SW_SHOW)  
Sleep(2000)
```

Ahora finalizamos el código utilizando un Lazo de Mensajes (MessageLoop) con la descripción de algunos mensajes de eventos. La declaración **Select...Case...EndSelect** nos permite tomar una decisión de acuerdo al evento "atrapado" por GUIGetMsg().

```
#include <GUIConstants.au3>  
GUICreate("Hola Mundo", 200, 100)
```

```

GUICtrlCreateLabel("Hola Mundo! Cómo estás?", 30, 10)
$okbutton = GUICtrlCreateButton("OK", 70, 50, 60)
GUISetState(@SW_SHOW)

While 1
$msg = GUIGetMsg()
Select
Case $msg = $okbutton ;evento de control
MsgBox(0, "Evento GUI", "Usted presionó OK!")
Case $msg = $GUI_EVENT_CLOSE ;evento del sistema
MsgBox(0, "Evento GUI", "Usted clickeó CLOSE! Finalizando...")
ExitLoop
EndSelect
WEnd

```

Esto resulta muy simple. Obviamente al crearse más ventanas y controles resulta más complicado pero el mecanismo siempre es el mismo, aquí solo mostramos lo básico.

GUIGetMsg Avanzado con Múltiples Ventanas

El ID de Control es único, aun cuando tenemos múltiples ventanas, de esta forma el código dado anteriormente trabaja bien con controles y ventanas múltiples. Sin embargo, cuando procesamos un evento como \$GUI_EVENT_CLOSE o \$GUI_MOUSEMOVE usted necesita conocer cual ventana GUI ha generado el evento. Para lograr esto usted debe llamar a GUIGetMsg de esta forma:

```
$msg = GUIGetMsg(1)
```

Cuando es llamada con el parámetro **1** el evento es devuelto en un valor de **arreglo** , el arreglo contiene el evento (en \$array[0]) e información extra como el apuntador de ventana (window handle) (en \$array[1]). Si dos ventanas fueran creadas en el ejemplo anterior entonces la manera correcta de escribir el código sería:

```

#include <GUIConstants.au3>
mainwindow = GUICreate("Hola Mundo", 200, 100)
GUICtrlCreateLabel("Hola Mundo! Cómo estás?", 30, 10)
$okbutton = GUICtrlCreateButton("OK", 70, 50, 60)
$dummywindow = GUICreate("Ventana de prueba ", 200, 100)
GUISwitch($mainwindow)
GUISetState(@SW_SHOW)
While 1

```

```

$msg = GUIGetMsg(1)
Select
Case $msg[0] = $okbutton
    MsgBox(0, "GUI Event", "Usted presionó OK!")
Case $msg[0] = $GUI_EVENT_CLOSE And $msg[1] = $mainwindow
    MsgBox(0, "Evento GUI", "Usted clickeó CLOSE en la ventana principal! Finalizando... ")
    ExitLoop
EndSelect
WEnd

```

El primer cambio principal que se puede notar es la llamada a la función [GUISwitch](#) - cuando una nueva ventana es creada esta se convierte en la ventana por "defecto" para futuras operaciones GUI (incluyendo la creación de controles). En nuestro caso deseamos trabajar con la ventana principal "Hola Mundo", no con la ventana de prueba, entonces cambiamos. Algunas funciones de GUI permiten que se use los apuntadores (handle) de la ventana en si misma- estas funciones permitirán conmutar de ventana automáticamente. En nuestro ejemplo esto sería de esta forma:

```
GUISetState(@SW_SHOW, $mainwindow)
```

El siguiente cambio a notar es el modo como se llama a la función GUIGetMsg y cómo se interpreta los eventos- note el uso de \$msg[0] y \$msg[1] - ahora solo salimos del script si el evento close es enviado **y** el evento es desde la ventana principal ("Hola Mundo").

Referencia GUI - Modo OnEvent

El modo OnEvent en vez de estar constantemente chequeando la GUI para reaccionar ante cualquier evento hace que la GUI temporalmente pause el script y llama a una función definida por usuario para manejar el evento. Por ejemplo, si usted hace click en el Botón1 la GUI pausa su script principal y llama a una función previa definida por usuario para tratar al Botón1. Cuando la llamada a la función es completada el script se resume. Este modo es similar a modo de chequeo de evento en Visual Basic.

Mientras la GUI está en ejecución, su script principal podría realizar cualquier trabajo normal, pero para facilidad de ejemplo haremos un script principal en "idles" (bajo consumo de CPU), en un ciclo While infinito.

El modo por defecto es el Lazo de Mensajes (MessageLoop) de esta forma si queremos utilizar el modo OnEvent tenemos que definir antes: [Opt\("GUIOnEventMode", 1\)](#).

Nota: Observe algo importante: La GUI trabaja de manera "independiente" del script en sí. La GUI puede estar a la espera de un evento mientras el script puede realizar complejas operaciones.

Formato Básico de OnEvent

El código general del modo OnEvent es el siguiente:

While 1

Sleep(1000) ;Idle (bajo consumo de CPU)

WEnd

Func Event1()

; Coloque aquí el código que dará respuesta el evento

EndFunc

Func Event2()

; Coloque aquí el código que dará respuesta el evento

EndFunc

Eventos GUIs

En el modo OnEvent la GUI puede generar los siguientes "Eventos":

Eventos de Controles

Eventos del Sistema

Ambos tipos de Evento harán la llamada a una función definida por usuario (UDF) ya sea para GUI ([GUISetOnEvent](#)) o para un control ([GUICtrlSetOnEvent](#)). Si ninguna función es definida

para un evento entonces simplemente se ignora. Existen varios macros que podemos utilizar dentro de la función llamada (UDF) los cuales nos dan ciertas informaciones útiles.

Macro	Detalles
@GUI_CTRLID	El ID de Control del control que envía el mensaje O el ID del evento del Sistema
@GUI_WINHANDLE	El Apuntador (handle) de la GUI que envía el mensaje
@GUI_CTRLHANDLE	El apuntador (Handle) que envía el mensaje (si es aplicable)

Nota: es perfectamente legal utilizar la misma función para acontecimientos múltiples, todos lo que usted necesita en estos casos puede tomarse en la información de la macro @GUI_CTRLID. Por ejemplo, usted puede registrar todos los eventos del sistema en alguna función.

Eventos de Controles

Cuando un control es clickeado o cambia el evento del control es enviado. El evento es atendido por la función definida con [GUICtrlSetOnEvent](#). Dentro de la función definida por usuario, de forma transparente, @GUI_CTRLID es fijado para **control ID** que es devuelto por la función [GUICtrlCreate](#)... cuando es creado.

Eventos del Sistema

Eventos del sistema - Como cuando es cerrada la GUI - son similares a los eventos de los controles, pero el tipo de evento es definido por @GUI_CTRLID. El evento es enviado a la función definida con [GUISetOnEvent](#). Los eventos posibles del sistema son los siguientes:

```
$GUI_EVENT_CLOSE      ;La GUI ha sido cerrada
$GUI_EVENT_MINIMIZE   ;La GUI ha sido minimizada
$GUI_EVENT_RESTORE    ; La GUI ha sido restaurada
$GUI_EVENT_MAXIMIZE   ; La GUI ha sido maximizada
$GUI_EVENT_PRIMARYDOWN ;Se ha oprimido el botón primario del mouse
$GUI_EVENT_PRIMARYUP    ;Se ha liberado el botón primario del mouse
$GUI_EVENT_SECONDARYDOWN ;Se ha oprimido el botón secundario del mouse
$GUI_EVENT_SECONDARYUP   ;Se ha liberado el botón secundario del mouse
$GUI_EVENT_MOUSEMOVE     ; Se ha movido el mouse
$GUI_EVENT_RESIZED       ; Ha cambiado de tamaño la GUI
$GUI_EVENT_DROPPED        ; Se ha soltado un objeto
```

Ejemplos de GUI

En la página principal de la [Referencia GUI](#) iniciamos un simple ejemplo de "Hola Mundo" que puede verse aquí:

```
#include <GUIConstants.au3>
GUICreate("Hola Mundo", 200, 100)
GUICtrlCreateLabel("Hola Mundo! Cómo estás?", 30, 10)
GUICtrlCreateButton("OK", 70, 50, 60)
GUISetState(@SW_SHOW)
Sleep(2000)
```

Ahora finalizamos el código usando OnEvents y algunos de los mensajes de eventos definidos más arriba.

```
#include <GUIConstants.au3>
Opt("GUIOnEventMode", 1) ; cambio para el modo OnEvent
$mainwindow = GUICreate("Hola Mundo", 200, 100)
GUISetOnEvent($GUI_EVENT_CLOSE, "Click en Close")
GUICtrlCreateLabel("Hola Mundo! Cómo estás?", 30, 10)
$okbutton = GUICtrlCreateButton("OK", 70, 50, 60)
GUICtrlSetOnEvent($okbutton, "OKButton")
GUISetState(@SW_SHOW)

While 1
    Sleep(1000)
WEnd

Func OKButton()
    ;Nota: en este punto @GUI_CTRLID es igual a $okbutton,
    ;y @GUI_WINHANDLE es igual a $mainwindow
    MsgBox(0, "Evento GUI", "Usted presionó OK!")
EndFunc

Func CLOSEClicked()
    ;Nota: en este punto @GUI_CTRLID será igual a $GUI_EVENT_CLOSE,
    ;y @GUI_WINHANDLE será igual a $mainwindow
    MsgBox(0, "Evento GUI", "Usted Clickeó CLOSE ! Finalizando...")
Exit
EndFunc
```

Esto resulta simple. Obviamente al crearse más ventanas y controles resulta más complicado pero el mecanismo siempre es el mismo, aquí solo mostramos lo básico.

Operaciones avanzadas y múltiples ventanas

El ID de control es único siempre que se tenga múltiples ventanas, pero cómo hacer referencia (handle) a cada ventana.

Aquí se muestra un ejemplo similar en el que existe una ventana principal y otra ventana hijo.

```
#include <GUIConstants.au3>

Opt("GUIOnEventMode", 1) ; Cambio al modo OnEvent
$mainwindow = GUICreate("Hola Mundo", 200, 100)
GUISetOnEvent($GUI_EVENT_CLOSE, "CLOSEClicked")
GUICtrlCreateLabel("Hola Mundo! Cómo estás?", 30, 10)
$okbutton = GUICtrlCreateButton("OK", 70, 50, 60)
GUICtrlSetOnEvent($okbutton, "OKButton")
$dummywindow = GUICreate("Ventana dummy para prueba ", 200, 100)
GUISetOnEvent($GUI_EVENT_CLOSE, "CLOSEClicked")
GUISwitch($mainwindow)
GUISetState(@SW_SHOW)
While 1
    Sleep(1000)
WEnd
Func OKButton()
    ;Nota: en este punto @GUI_CTRLID será igual a $okbutton
    MsgBox(0, "GUI Event", "Usted presionó OK!")
EndFunc
Func CLOSEClicked()
    ;Nota: en este punto @GUI_CTRLID será igual a $GUI_EVENT_CLOSE,
    ;@GUI_WINHANDLE es semejante a $mainwindow o $dummywindow
    If @GUI_WINHANDLE = $mainwindow Then
        MsgBox(0, "GUI Event", "Usted cliqueó CLOSE en la ventana principal! finalizando...")
        Exit
    EndIf
EndFunc
```

El primer cambio que puede notarse es la llamada a la función [GUISwitch](#) - cuando se crea una ventana nueva esta se convierte en la ventana por "defecto" para futuras operaciones GUI (incluyendo creación de controles). En nuestro caso deseamos trabajar con la ventana principal "Hola Mundo", no con la ventana de prueba, entonces hacemos una(commutación)switch. Algunas funciones GUI permiten el uso del apuntador de la ventana

(handle) dentro de la misma función - estas funciones harán la conmutación automáticamente. En nuestro ejemplo podríamos hacer esto con:

```
GUISetState(@SW_SHOW, $mainwindow)
```

También puede notarse que utilizamos la misma función OnEvent para el apuntador (handle) el botón "close" para ambas ventanas y después usamos @GUI_WINHANDLE para determinar que ventana envió el mensaje - entonces solo se cierra la GUI cuando el botón CLOSE es clickeado **y el mensaje es llamado desde la ventana principal**. Usted puede fácilmente separar las funciones de cada ventana si lo deseas.

Observe que todas las declaraciones actúan de manera muy similar, son apenas diferentes y cada una puede ser más apropiada que la otra, dependiendo de la situación específica que usted presente.

REFERENCIA DE OBJETOS COM

De las Extensiones COM al código para AutoIt

Una breve introducción

Que es COM?

Entiéndase COM, como un "Modelo de Objeto Componente". Es el método de Microsoft para interconectar software usando una interfaz común. Esas interfaces se encuentran definidas en un Objeto COM.

Antes de la existencia de COM, usted debía conocer la implementación exacta de un programa antes de poder interactuar con él. Usando COM, tiene ahora la capacidad de "hablarle" a estos Objetos definidos. Sin embargo, debe tener conocimiento de los nombres de los Objetos que se utilizan y cuáles son sus 'propiedades' o los 'métodos' disponibles.

¿Que son propiedades o métodos (de un Objeto)?

Hay dos características básicas para un objeto. Podría verse una 'propiedad' como el almacenamiento de datos para un Objeto dado. Y un 'método' puede ser visto como una función interna llamada a hacer algo con esos datos.

¿Se necesita COM en un script de AutoIt?

Simplemente depende. AutoIt posee muchísimas funciones incorporadas y una amplia librería de UDFs. La mayor parte de la programación puede realizarse con ellas. Sin embargo, si usted requiere un 'enlace' especial a otras aplicaciones, usando COM debería ahorrar algunas líneas de script. Los desarrolladores de código deben ser conscientes que la existencia de un Objeto COM depende EN GRAN MEDIDA del sistema operativo y el software instalado. Los ejemplos citados abajo fueron todos probados en una versión 'plana' de Windows XP professional con Microsoft Office 2000.

Un ejemplo de uso con COM en AutoIt

Digamos que se desea minimizar todas las ventanas abiertas. Usted podría hacerlo usando funciones regulares de AutoIt tales como WinList y WinsetState. Sin embargo, dos líneas de código COM pueden proporcionar el mismo resultado:

```
$oShell = ObjCreate("shell.application")
```

```
$oShell.MinimizeAll
```

Nota al margen: El ejemplo ya no es la manera más breve de minimizar todas las ventanas, luego de la incorporación de la función WinMinimizeAll() en AutoIt.

En la primera línea, creamos un nuevo objeto llamado "shell.application". Es un Objeto interno de Windows, definido en shell32.dll. El puntero a este nuevo objeto se asigna a la variable \$oShell. \$oShell es desde ahora una variable Objeto.

En la segunda línea, usamos un Método llamado "MinimizeAll" al objeto oShell. Este minimizará todas las ventanas.

Todas las versiones de Windows tienen una amplia cantidad de Objetos internos para propósitos variados. Y aplicaciones como Excel o Word también tienen su propio conjunto de Objetos.

No obstante, en algunas circunstancias se torna complejo obtener una lista de todos los Objetos definidos existentes en el sistema, con sus correspondientes Propiedades y Métodos. Buscar por Microsoft.com o Google.com podría proporcionarle alguna pista acerca del Objeto 'X' que usted pretende utilizar.

Por ejemplo, puede encontrar información sobre el objeto "shell.application" en:

<http://msdn.microsoft.com/en-us/library/bb774094.aspx>

Para conseguir un acercamiento a todos los objetos actualmente instalados en su sistema, el "Visor de Objetos OLE/COM" es una muy útil herramienta de ayuda. Más adelante se ampliará sobre esta herramienta en una sección aparte.

Ahora otro ejemplo. Supongamos que se desea capturar el código fuente HTML de cierta página web. Usted podría usar la función interna InetGet() para guardar el archivo resultante y luego leerlo a través de FileRead(). Pero estas líneas de código hacen lo mismo:

```
$oHTTP = ObjCreate("winhttp.winhttprequest.5.1")
$oHTTP.Open("GET", "http://www.AutoItScript.com")
$oHTTP.Send()
$HTMLSource = $oHTTP.Responsetext
```

La variable (de cadena) \$HTMLSource ahora contiene el código HTML completo para la página de inicio de AutoItScript.com (Este sería, el cuadro HTML superior).

(La información sobre el objeto "winhttp.winhttprequest" puede encontrarse en:
<http://msdn.microsoft.com/en-us/library/aa384106.aspx>)

Por favor recuerde: La existencia de Objetos depende del sistema operativo de la computadora y los programas instalados. Por ejemplo, el objeto winhttp.winhttprequest.5.1 solamente existe en computadoras que tienen al menos la versión 5 de Internet Explorer instalada. Cuando usted comparte scripts que usan objetos COM, asegúrese que los objetos existan en esas computadoras.

Las variables que contienen Objetos se comportan algo diferente en relación a otros tipos de variables en Autolt. Un Objeto no es realmente un valor, sino un 'puntero' a algo externo al script. Por lo tanto no puede realizar aritmética ni ecuaciones con variables del tipo Objeto. Cuando le asigne a una variable Objeto un valor diferente, el 'puntero' se liberará automáticamente. Usted puede, por ejemplo, forzar el borrado de un Objeto, asignándole cualquier valor numérico o de texto.

```
$oHTTP = ObjCreate("winhttp.winhttprequest.5.1") ; El objeto es creado  
$oHTTP=0 ;El objeto es borrado
```

No es **necesario** borrar los Objetos cuando está finalizando. Si un script termina, Autolt intentará soltar todas las referencias activas a los Objetos que han sido creados en el script. Lo mismo sucede con una variable de entorno local, si se definió un Objeto dentro de una función, y esta termina con return.

Automatización usando COM

Una aplicación muy popular del COM es la de 'automatizar' programas. En lugar de usar las funciones regulares de Autolt, como lo son Send() o WinActivate(), usted puede hacer uso de Objetos que están definidos dentro del programa.

Aquí un ejemplo para 'automatizar' el Microsoft Excel:

```
$oExcel = ObjCreate("Excel.Application")           ; Crea un Objeto Excel  
$oExcel.Visible = 1                            ; Permite mostrarle el Excel a usted  
$oExcel.WorkBooks.Add                          ; Agrega un nuevo libro de trabajo  
$oExcel.ActiveWorkBook.ActiveSheet.Cells(1,1).Value="prueba" ; LLena una celda  
sleep(4000)                                     ; Ver los resultados durante 4 segundos  
$oExcel.ActiveWorkBook.Saved = 1                ; Guardado simulado del libro  
$oExcel.Quit                                    ; Cierra el Excel
```

La complejidad para operar con otros programas dependerá específicamente del programa en cuestión, y no del script Autolt. Si algo no funciona como usted esperaba, debería intentar consultar la documentación para 'esa aplicación' en vez de la ayuda de Autolt.

Declaraciones especiales

En Autolt, existen dos declaraciones especiales que fueron diseñadas para el uso con COM:

WITH/ENDWITH y el bucle FOR/IN/NEXT.

WITH..ENDWITH

La declaración WITH/ENDWITH no agrega funcionalidad, pero le permite escribir código y que este se interprete de una manera más sencilla. Así, el anterior ejemplo de uso con Excel, pudiera ser escrito también como sigue:

```
$oExcel = ObjCreate("Excel.Application") ; Crea un Objeto Excel  
  
WITH $oExcel  
    .Visible = 1 ; Permite mostrarle el Excel a usted  
    .WorkBooks.Add ; Agrega un nuevo libro de trabajo  
    .ActiveWorkBook.ActiveSheet.Cells(1,1).Value="Prueba" ; Llena una celda  
    sleep(4000) ; Le muestra los resultados 4 segundos  
    .ActiveWorkBook.Saved = 1 ; Guardado simulado del libro de trabajo  
    .Quit ; Cierra el Excel  
ENDWITH
```

Este ejemplo en particular no resume considerablemente el texto, pero cuando en un objeto usted use abundantes líneas de propiedades/métodos, podrá acortar trabajo de forma tangible usando la declaración WITH.

FOR..IN

El bucle FOR...IN se requiere cuando usa Colecciones. Las Colecciones son un tipo especial de Objeto, que existe fuera de múltiples sub-Objetos. Podríamos compararlos con los Arreglos (de hecho, actualmente la declaración FOR..IN también funciona con variables del tipo Arreglo).

Bucle FOR..IN usando un Arreglo

Debajo hay un ejemplo de un bucle FOR..IN. Este ejemplo usa un arreglo normal de AutoIt, por lo tanto nada tiene de un COM. Sólo es con el fin de mostrarle el principio de uso:

```
$String = "" ; Una cadena para mostrarle el propósito  
  
$aArray[0]="a" ; Llenamos un arreglo  
$aArray[1]=0 ; con algunos  
$aArray[2]=1.3434 ; diferentes  
$aArray[3]="testestestest" ; valores de ejemplo.  
  
FOR $Element IN $aArray ; Aquí comienza...  
    $String = $String & $Element & @CRLF  
NEXT
```

```
; Ahora muestra los resultados al usuario  
Msgbox(0,"Prueba Arreglo For..IN","Resultado: " & @CRLF & $String)
```

Bucle FOR..IN usando un Objeto

En muchos casos usted no puede utilizar métodos 'normales' de Objeto para obtener los elementos que conforman una Colección. En términos 'COM' ellos le dicen a usted como se 'enumera' la serie. Aquí es donde el bucle FOR..IN entra en escena.

En el ejemplo de Excel abajo un bucle de celdas A1:O16 sobre la actual hoja activa. Si una de las celdas tiene un valor por debajo de 5, el código efectuará un reemplazo con 0 (cero):

```
$oExcel = ObjCreate("Excel.Application") ; Crea un Objeto Excel  
  
$oExcel.Visible = 1 ; Permite mostrarle el Excel a usted  
$oExcel.WorkBooks.Add ; Agrega un nuevo libro de trabajo  
  
dim $arr[16][16] ; Estas lineas  
for $i = 0 to 15 ; son simplemente  
    for $j = 0 to 15 ; un ejemplo para  
        $arr[$i][$j] = $i ; crear algo de  
    next ; contenido para las celdas.  
next  
  
$oExcel.activesheet.range("A1:O16").value = $arr ; LLena celdas con números de ejemplo  
  
sleep(2000) ; Aguarda un momento antes de continuar  
  
For $cell in $oExcel.ActiveSheet.Range("A1:O16")  
    If $cell.Value < 5 Then  
        $cell.Value = 0  
    Endif  
Next  
  
$oExcel.ActiveWorkBook.Saved = 1 ; Guardado simulado del libro de trabajo  
sleep(2000) ; Aguarda un momento antes de continuar  
$oExcel.Quit ; Cierra Excel
```

Uso avanzado de COM

Las siguientes características de AutoItCOM requieren un conocimiento profundo de los Eventos COM y de control para captura de Errores COM.

Si usted recién se está iniciando en este tipo de programación, por favor lea antes documentación algo más simple referida a COM.

La Biblia de COM es un libro que tiene por nombre "Inside OLE 2" (Dentro del OLE 2) por Kraig Brockschmidt (Microsoft Press).

También puede encontrar recursos COM en internet (no relacionados con AutoIt):

<http://msdn.microsoft.com/en-us/library/ms694363.aspx> (introducción)

<http://www.garybeene.com/vb/tut-obj.htm> (sobre Objetos en Visual Basic)

<http://java.sun.com/docs/books/tutorial/java/concepts/> (Usando objetos en Java)

<http://msdn.microsoft.com/archive/en-us/dnarguion/html/drgui082399.asp> (Eventos de Objeto en C++)

<http://www.garybeene.com/vb/tut-err.htm> (Captura de errores en Visual Basic)

Eventos COM

Por lo general, la Automatización COM usa principalmente comunicación en un único sentido. Usted 'solicita' al Objeto mediante cualquier propiedad o resultado a partir de un método. Sin embargo, un Objeto COM puede en ocasiones 'devolver una respuesta' a su script.

Podría ser muy oportuno en caso de que usted necesite esperar que suceda alguna acción relacionada al COM.

En lugar de escribir cantidades de bucles, pregunte al Objeto si algo de su interés sucedió. Usted puede delegar esa tarea para que el Objeto llame a una UDF específica en su script. Entre tanto usted puede ejecutar otras cosas en su código (casi) simultáneamente.

No todos los Objetos soportan Eventos. Lea la documentación sobre el objeto cuidadosamente para determinar si los soporta o no.

En caso de comprobar que sí, el segundo paso es lograr determinar qué tipos. AutoItCOM puede recibir eventos del tipo 'dispatch' solamente.

Para concluir, usted debe conocer los nombres de los Eventos que el Objeto podría generar, incluyendo sus argumentos (si los hubiere).

Solo cuando disponga de esta información, usted podrá comenzar a componer un script AutoIt que ejecute Eventos COM.

Deabajo hay un fragmento del código de un script que demuestra como recibir Eventos desde el Internet Explorer:

```
$oIE=ObjCreate("InternetExplorer.Application.1") ; Crea un Objeto de Internet Explorer
```

```
$EventObject=ObjEvent($oIE,"IEvent_","DWebBrowserEvents") ; Comienza la recepción de Eventos.
```

*\$oIE.url= "http://www.autoitscript.com" ; Abre una página Web de ejemplo

;Partiendo desde aquí, el Objeto \$oIE genera eventos durante la carga del sitio.
;Ellos son capturados en las funciones de evento mostradas más abajo.*

*;Aquí usted puede hacer que el script permanezca esperando hasta que el usuario finalice.
...(su código estaría aquí)...*

```
$EventObject.stop      ; Nosotros detenemos la captura de Eventos IE ahora  
$EventObject=0        ; Liberamos los Eventos para el Objeto  
$oIE.quit            ; Cierra IE  
$oIE=0               ; Remueve IE desde la memoria (no es necesario)  
Exit                 ; Fin del script principal
```

*; Unas pocas Funciones de Evento para Internet Explorer.
;
; Para una lista completa de Funciones de Evento IE, vea la documentación de MSDN
Navegador web en:
; <http://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser.aspx>*

```
Func IEEVENT_StatusTextChange($Text)  
; En el script completo (vea el enlace debajo) le mostramos el contenido en una caja GUI de  
Edición.  
    GUICtrlSetData ( $GUIEdit, "Estado del texto IE cambió a: " & $Text & @CRLF, "append" )  
EndFunc
```

```
Func IEEVENT_BeforeNavigate($URL, $Flags, $TargetFrameName, $postData, $Headers,  
$Cancel)  
; En el script completo (vea el enlace debajo) le mostramos el contenido en una caja GUI de  
Edición.  
; Nota: la declaración es diferente de la que figura en el sitio MSDN.  
    GUICtrlSetData ( $GUIEdit, "AntesNavegar: " & $URL & " Bandera: " & $Flags &  
@CRLF, "append" )  
EndFunc
```

Click [aquí](#) para ver el script completo.

La línea principal en el script es:

```
$EventObject=ObjEvent($oIE,"IEEvent_",__.);
```

Esta función toma el objeto \$oIE y redirecciona estos eventos a las funciones de AutoIt cuyos nombres comienzan con MYEvent_. El tercer parámetro es opcional. Se emplea cuando un Objeto tiene múltiples interfaces de Evento y usted no desea que AutoIt las seleccione automáticamente.

El Objeto responsable del redireccionamiento continuo es \$EventObject. Esta variable no requiere de una atención importante, a menos que quiera detener la captura de eventos.

Para terminar con la redirección, no basta con borrar la variable con \$EventObject="" . puesto que el Objeto todavía mantiene el 'llamado' con referencia a esta variable, y no la perderá sino hasta que el Objeto mismo la cierre. Usted podría resolver este inconveniente borrando el Objeto que está 'llamando'. Sin embargo es recomendable hacerlo diciendo al Objeto que ya no desea recibir los Eventos: \$EventObject.Stop. Luego puede (sin ser realmente necesario) limpiar el Evento asignándole cualquier valor, como: \$EventObject=""

Si usted conoce los nombres de los Eventos que \$oIE alerta, puede implementar los que a usted le interesan creando UDFs para Autolt con los nombres IE Event_NombreEvento(*argumentos opcionales*). Asegúrese de estar usando el número correcto de argumentos y en el orden correspondiente, como se especifica para esa función de evento. De otra manera usted puede obtener algunos valores inesperados.

Si usted no conoce (por alguna razón) los nombres de los eventos, puede agregar una UDF con el prefijo únicamente. Por ejemplo: Func IEEVENT_(*\$Eventname*).

Cuando un evento es recibido sin que IEEVENT_ *NombreEvento* UDF exista, esta función será llamada en su lugar y el nombre del evento será guardado en la variable *\$Eventname*.

No es requerido implementar TODAS las funciones de evento. Aquellas que no lo estén, sencillamente serán ignoradas.

Existen mas scripts de ejemplo que usan funciones de Evento COM en el directorio test para la versión BETA de Autolt, descargue el archivo ZIP distribuible desde: <http://www.autoitscript.com/autoit3/files/beta/autoit/COM/>

Limitaciones con Eventos COM en Autolt

Algunos Objetos (como el 'NavegadorWeb') transfiere argumentos a sus Funciones de Evento 'por referencia'. Esto tiene como objetivo permitir al usuario cambiar esos argumentos y variables devolviéndolas al Objeto. Sin embargo, Autolt usa su propio juego de variables , el cual no es compatible con las variables COM. Significa que todos los valores desde Objetos necesitan ser convertidos en variables de Autolt, perdiendo así la referencia al espacio de memoria que se les destinó originalmente. Es probable que en un futuro cercano podamos resolverle a usted esta limitación!

Captura de Errores COM

Usar COM sin una apropiada captura de errores, suele traer muchas complicaciones. Especialmente cuando no está familiarizado con el uso de Objetos en su script.

Un script de Autolt detendrá inmediatamente su ejecución cuando detecte un error COM . Esta configuración se encuentra predefinida y de hecho es la más segura. Siendo este el caso, deberá tomar las medidas necesarias para **prevenir** que un error ocurra.

Solamente si no hay manera de **prevenir** un error COM, podría emplazar un "Capturador de Errores" en el cual usted tome una acción **después** de que el error sucedió. Esta **no** es la solución propuesta de seguimiento para hacer que el script trabaje correctamente. Ni tampoco está destinada a capturar errores no relativos al COM (ej: declaraciones y errores en la sintaxis).

La captura de Errores se implementa en la misma forma que un Evento COM normal, usando ObjEvent() y el usuario define entonces la Función Evento. La única diferencia es el uso fijo de la cadena "Autolt.Error" en el lugar del nombre del Objeto.

Un ejemplo:

\$oMyError = ObjEvent("Autolt.Error","MyErrFunc") ; Emplaza un capturador de errores propio.

```
; Aquí se lleva a cabo un fallo deliberado (el objeto no existe)
$olE = ObjCreate("InternetExplorer.Application")
$olE.Visible = 1
$olE.Inventado
if @error then MsgBox(0,"","la linea anterior tiene un error.")
```

Exit

```
; Este es mi capturador de errores propio
Func MyErrFunc()
    $HexNumber=hex($oMyError.number,8)
    MsgBox(0,"","Se interceptó un error COM!" & @CRLF &_
        "El número es: " & $HexNumber & @CRLF &_
        "La descripción obtenida es: " & $oMyError.WindowDescription )
```

```
setError(1) ; algo que colocamos para controlar cuando la función retorna
Endfunc
```

Acerca del Capturador de Errores de Evento, algo especial es lo que el Objeto retorna. Este es un Objeto de Error en Autolt que contiene algunas propiedades y métodos útiles. Su implementación está parcialmente basada en el Objeto "Err" de VB(script):

Propiedades del Objeto Autolt Error:

.number	El valor Windows HRESULT de un llamado COM
.windescription	El texto FormatWinError() derivado desde .number
.source	Nombre del Objeto que generó el error (contenidos desde ExceInfo.source)

.description	La descripción del Objeto que originó el error (contenido desde ExcepInfo.description)
.helpfile	Archivo de ayuda del Objeto que originó el error (contenido desde ExcepInfo.helpfile)
.helpcontext	Número Id contextual para el archivo de ayuda del Objeto (contenido desde ExcepInfo.helpcontext)
.lastdllerror	El número de retorno desde GetLastError()
.scriptline	La linea del script en la cual el error fue generado

Métodos del Objeto Autolt Error:

.raise	Promueve un Evento de error, iniciado por el usuario
.clear	Limpia el contenido del Objeto Error (Ej: número a 0, cadena en "")

Nota para los desarrolladores de UDF

Unicamente puede disponer de UN Capturador de Errores de Evento por Script de Autolt. Si está escribiendo UDFs con contenido COM, para revisar si el usuario emplazó un Capturador de Errores propio, proceda como sigue:

```
$sFuncName = ObjEvent("Autolt.Error")
```

```
if $sFuncName <> "" then MsgBox (0,"Prueba","El usuario ya emplazó una función de Captura de Errores: " & $sFuncName)
```

En caso de que no hubiera un Capturador de Errores activo, puede entonces emplazar temporalmente el suyo durante el curso al llamado de la UDF.

Sin embargo, nunca podrá detener uno ya existente sin liberar antes la variable que le fue asignada al mismo. Si el autor del script hubiera emplazado un Capturador de Errores COM, es responsabilidad de aquel utilizar una función apropiada que pueda contemplar también errores COM generados por las UDFs.

El Visor de Objetos OLE/COM

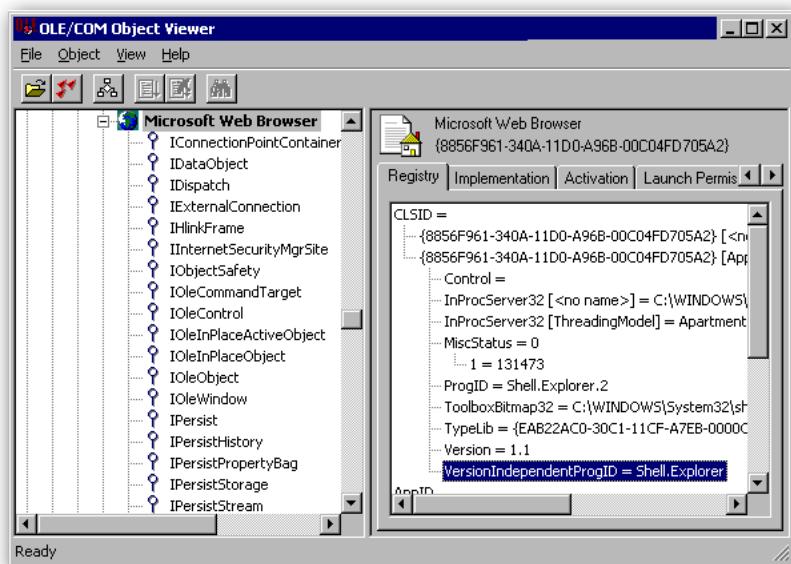
En inglés llamada "OLE/COM Object Viewer", es una herramienta muy conveniente para el objetivo de espiar en todos los objetos COM actualmente instalados en el sistema. Es una

parte del Windows 2000 resource kit y puede ser descargado gratuitamente en:
<http://www.microsoft.com/downloads/details.aspx?familyid=5233b70d-d9b2-4cb5-aeb6-45664be858b6&displaylang=en>

El instalador del programa es algo rústico. No le creará ningún ícono de acceso en el Menú Inicio. En su lugar, se instalará el archivo oleview.exe en el directorio C:\Archivos de Programa\Resource Kit (instalación predeterminada).

Cuando corra el oleview.exe, algunos sistemas alertarán sobre un archivo faltante con nombre iviewers.dll. Este archivo es requerido, pero contrariamente a lo que uno podría suponer lógico, no se encuentra incluido en el instalador. Puede obtener este dll desde una versión antigua del oleview.exe en: <http://download.microsoft.com/download/2/f/1/2f15a59b-6cd7-467b-8ff2-f162c3932235/ovi386.exe>. Este paquete ahora instalará los archivos de manera predeterminada en la ruta C:\MSTOOLS\BIN. Como usted únicamente necesita el archivo iviewer.dll, copie el mismo al directorio donde oleview.exe se localiza, y luego registre el dll usando la linea de comandos: regsvr32 iviewers.dll.

Veamos ahora un ejemplo con el Oleviewer. Ejecutelo y siga esta rama: Object Classes->Grouped by Component Category->Control->Microsoft Web Browser.

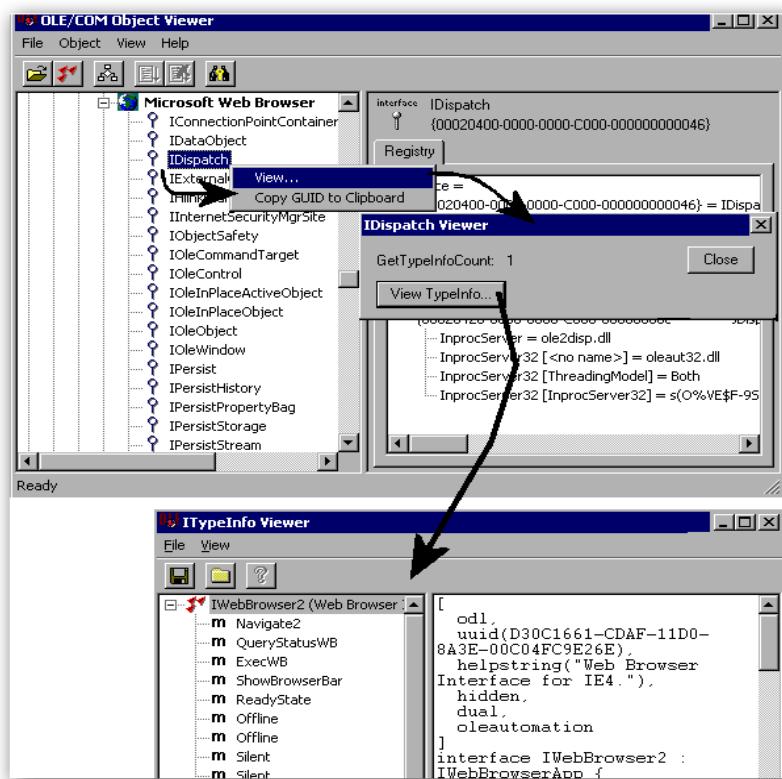


En la columna a la izquierda usted verá todas las interfaces COM que han sido definidas para este objeto. Hablaremos de ella más tarde. A continuación observe con detenimiento la columna de la derecha. Esta contiene mucha información para el uso de este objeto en un script de AutoIt. Lo más importante es "VersionIndependentProgID". Este será el nombre a emplear con la función ObjCreate, ObjGet u ObjEvent. Además contiene el directorio y nombre del archivo para el objeto. Pudiera ser un EXE, un DLL o un archivo OCX. InProcServer32

significa que el objeto corre en el mismo hilo que su script (en-proceso). Cuando visualice en cambio LocalServer32, sepa que el objeto correrá como un proceso separado. El objeto debe también contener un tipo de librería (la linea a continuación de "TypeLib="), de lo contrario no es posible utilizarlo en un script de AutoIt.

Las interfaces para la columna ubicada a la izquierda son usadas de diversas maneras a los fines de interactuar con el objeto. Algunas son empleadas para almacenamiento (IStorage, IPersist), otras para insertarlas en un GUI (IOleObject, IOleControl). Son los usos en AutoIt con IDispatch para la interfaz de automatización. La misma pone al 'descubierto' todos los métodos de escritura de código y propiedades que el objeto soporta. Si no existiesen, usted no podrá utilizar el objeto en un script de AutoIt.

A continuación centramos la mirada en esta interfaz. Click-derecho sobre el nombre IDispatch y seleccionamos "View..." desde el menú contextual. Ahora click en el botón "View TypeInfo..." . (Nota: si el mismo aparece inhabilitado, sucedió que no tiene registrado el archivo iviewers.dll , o el objeto no tiene una librería del tipo.)



La ventana "ITypeInfo Viewer" solamente mostrará la información que proporciona el objeto. Si el desarrollador decide no incluir un archivo de ayuda, usted verá únicamente los métodos/propiedades y nada más que eso. La librería "Microsoft Web Browser" de cualquier manera es bastante extensa. Haga click en un ítem de la columna izquierda, y entonces la descripción será mostrada a la derecha. En ocasiones tendrá que revisar las "Inherited Interfaces" para obtener más métodos para el objeto.

La sintaxis para los métodos/propiedades descriptas aquí figuran al estilo C/C++. Una propiedad descripta como "HRESULT Resizable([in] VARIANT_BOOL pbOffline)", deberá ser transcripta al lenguaje AutoIt de esta forma: \$Resizable=\$Object.Resizable (\$Object contiene el objeto creado con ObjCreate u ObjGet).

Términos propensos a confusión

Estos términos son por lo general confundidos con COM, pero su significado es diferente:

OOP = Programación Orientada a Objetos. Una técnica de programación con la cual los componentes del software son agrupados en bloques reutilizables conocidos como Objetos.

DDE = Intercambio dinámico de datos.

Podría decirse que es predecesor del COM. Lo usó IPC para transferir información y comandos entre aplicaciones diferentes.

OLE = Enlace al Objeto e Inclusión

En su primer versión, OLE fue una versión extendida del DDE para 'insertar' datos en un programa desde otro. La generación actual del OLE es establecimiento del COM y forma parte del ActiveX.

Automatización = Esta es una vía para manipulación de los Objetos en otras aplicaciones. Se utiliza con OLE, ActiveX y COM.

ActiveX = La siguiente generación OLE con Automatización, desarrollado desde el inicio con el propósito de servir de interfaz entre aplicaciones conectadas en una red funcional (especialmente navegadores web). ActiveX se diseña en base al COM.

DCOM = COM Distribuido. Una leve modificación del COM, que posibilita comunicación entre diferentes computadoras físicas.

.NET (punto Net) = No es realmente una porción de software, sería un 'concepto' ideado por Microsoft para interconectar simplemente de "todo" a través de (su) software. "punto Net" es empleado mayormente para servicios basados en Web.

COMunista = No es parte integrante del COM, sino alguien que cree en el comunismo (una teoría que sostiene que la gente ordinaria debería poseer toda la propiedad).

Referencia de Palabras Claves

Abajo se muestra una lista completa de las Palabras Reservadas en Autolt.

Keyword	Descripción
False / True	Valores boléanos para ser usados en expresiones lógicas.
#comments-start	Especifica que una sección entera en el script debe ser tomado como comentario.
ContinueCase	En un bloque Select o Switch, aborta el actual Case y continua en el siguiente.
ContinueLoop	Continua un ciclo While/Do/For.
Default	Palabra reservada usada dentro de llamadas a funciones.
Dim / Global / Local / Const	Declara una variable, una constante, o crea un arreglo.
Do...Until	Ciclo basado en una expresión.
Enum	Enumera constantes.
Exit	Termina la ejecución del script.
ExitLoop	Termina un ciclo While/Do/For
For...To...Step...Next	Ciclo basado en una expresión.
For...In...Next	Enumera elementos en un objeto de colección en un arreglo.
Func...Return...EndFunc	Define una Función de Usuario que toma cero o más argumentos y devuelve opcionalmente un resultado.
If...Then	Declaración condicional simple.
If...Elseif...Else...EndIf	Declaración condicional.
#include-once	Especifica que el fichero actual pueda establecerse una vez.
#include	Incluye archivos en el script actual.
#NoAutolt3Execute	Especifica que el actual script compilado no puede correr con los parámetros /Autolt3ExecuteLine o /Autolt3ExecuteScript.
#NoTrayIcon	Indica que el ícono de Autolt en la barra de tareas no será mostrado cuando el script sea ejecutado.
#OnAutoltStartRegister	Registra una función para ser llamada cuando Autolt inicia.
ReDim	Redimensiona un arreglo existente.
#RequireAdmin	Especifica que el actual script requiere derechos de administrador.
Select...Case...EndSelect	Declaración condicional

<u>Static</u>	Declara una variable estática o crea un arreglo estático.
<u>Switch...Case...EndSwitch</u>	Declaración condicional.
<u>While...WEnd</u>	Ciclo basado en una expresión.
<u>With...EndWith</u>	Usado para reducir referencias largas a variables de tipo objeto.

#comments-start

Especifica que una sección entera en el script debe ser tomado como comentario.

```
#comments-start
...
...
#comments-end
```

Parámetros

Ninguno

Comentarios

Las directivas #comments-start y #comments-end puede ser anidadas. Usted puede también usar las palabras claves abreviadas **#cs** y **#ce**. Adicionalmente, pueden ser comentadas!

Relativo

Ejemplo

```
#comments-start
MsgBox(4096, "", "Esto no será ejecutado")
MsgBox(4096, "", "o esto")
#comments-end

;;; #cs
MsgBox(4096, "", "Esto se imprime si '#cs' es quitado.")
#ce
```

#include

Incluye archivos en el script actual.

```
#include "[ruta]\nombre de fichero"
#include <nombre de fichero>
```

Parámetros

nombre de	El nombre del archivo del actual script para ser incluido. Ruta es opcional. Este
-----------	--

fichero	debe ser una cadena - No puede ser una variable. Si "..." es usado, el nombre de archivo es tomado como relativo al actual script. If <...> es usado el nombre de archivo es tomado como relativo al directorio include de la librería (usualmente C:\Program Files\AutoIt3\Include). La librería include contiene algunas funciones de usuario pre-escritas listas para su uso!
---------	---

Comentarios

En un script de AutoIt , otros scripts puede ser incluido usando el comando #include"

Si usted incluye un archivo con una función de usuario duplicada recibirá un error: "Duplicate function". Cuando se incluye un archivo, para asegurarse de que no contiene funciones duplicadas debe escribirse al principio una línea que contenga **#include-once** para prevenir el error.

Existe un valor especial en el registro que puede ser creado en la siguiente dirección: "HKEY_CURRENT_USER\Software\AutoIt v3\AutoIt" llamado "Include". Este de tipo REG_SZ (cadena). Este valor tiene una lista de directorio delimitados por "punto y coma" (;) para la localización de archivos cuando se utiliza #include's en adición con las localizaciones estándar.

El orden de búsqueda usado por AutoIt depende de cual forma de #include usted use. En la tabla de abajo se muestra el orden propuesto para la búsqueda.

Usando #include <>

Librería estándar	La ruta del intérprete actual en ejecución con "\Include" adicionado a la búsqueda.
Librería definida de usuario.	El valor de registro mencionado anteriormente es leído y cada directorio es buscado en el orden en que aparecen.
Directorio de script	El directorio desde el que se está ejecutando el script actualmente.

Usando #include "" (Constituye el inverso de #include <>).

Directorio de script	El directorio desde el que se está ejecutando el script actualmente.
Librería definida de usuario	El valor de registro mencionado anteriormente es leído y cada directorio es buscado en el orden en que aparecen.
Librería estándar	La ruta del intérprete actual en ejecución con "\Include" adicionado a la búsqueda.

Una nota acerca de uso de la opción /AutoIt3ExecuteScript. Puesto que la Librería estándar es buscada por el directorio actual del intérprete, Las funciones de la Librería estándar no serán encontradas; la Librería estándar solo puede ser encontrada ejecutando AutoIt3.exe. Es recomendable que usted compile sus scrip con el formato .a3x antes de intentar ejecutar el script utilizando /AutoIt3ExecuteScript.

Aut2Exe usa el mismo algoritmo que AutoIt3.exe con la sola diferencia que este busca el subdirectorio de include como si fuera el subdirectorio hermano de si mismo (..\Include).

Si Opt("TrayIconDebug",1) es fijado solamente nombres de ficheros de 64 include pueden ser mostrados en el traytooltip. Para otros nombres de archivos no será mostrado.

Relativo

#include-once

Ejemplo

```
;;; TIME.AU3 ;;
MsgBox(0, "", "La hora es" & @HOUR & ":" & @MIN & ":" & @SEC)
```

```
;;; SCRIPT.AU3 ;;
#include "TIME.AU3"
MsgBox(0, "", "Ejemplo")
#include "TIME.AU3"
Exit
```

; Ejecutar el script.au3 con tres mensajes de salida:
; uno con el tiempo, uno con: 'Example', y otro con el tiempo.

#include-once

Especifica que el fichero actual pueda establecerse una vez.

#include-once

Parámetros

Ninguno

Comentarios

Si usted include el mismo archivo conteniendo una función de usuario más de una vez, usted obtendrá un error "Duplicate function". Cuando escriba un archivo incluido que pueda estar usando de este modo, coloque en la línea superior: **#include-once** para prevenir que este pueda estar más de una vez.

Relativo

#include

Ejemplo

```
;;; LIBRARY.AU3 ;;
```

```

#include-once

Func myFunc()
    MsgBox(0, "", "Hola desde library.au3")
EndFunc

;; SCRIPT.AU3 ;;
#include "Library.au3"
#include "Library.au3" ;envía un error si #include-once no es usado

MsgBox(0, "Ejemplo", "Esto es desde el fichero 'script.au3'")
myFunc()
Exit

; Ejecutar el script.au3 con dos mensajes:
; uno diciendo "Esto es desde el fichero 'script.au3"
; y otro diciendo "Hola desde library.au3"

```

#NoTrayIcon

Indica que el icono de AutoIt en la barra de tareas no será mostrado cuando el script sea ejecutado.

#NoTrayIcon

Parámetros

Ninguno

Comentarios

Es posible utilizar Opt("TrayIconHide", 1) para remover el icono de la barra de tarea pero este será visible por un segundo mientras se inicia el script. Colocando la directiva #NoTrayIcon en la parte superior de su script se asegura de que el icono no sea mostrado. Usted puede también volver a mostrar el icono utilizando Opt("TrayIconHide", 0)

Relativo

[TrayIconHide \(Option\)](#)

Ejemplo

```

#NoTrayIcon
MsgBox(4096, "Click OK", "Muestra el icono de bandeja por 5 segundos... ")
Opt("TrayIconHide", 0) ;le quita el modo oculto del icono
Sleep(5000)

```

#OnAutoItStartRegister

Registra una función para ser llamada cuando AutoIt inicia.

```
#OnAutoItStartRegister "función"
```

Parámetros

función	El nombre de la función a llamar..
---------	------------------------------------

Comentarios

Ninguno.

Relativo

Ninguno.

Ejemplo

```
#OnAutoItStartRegister "MyTestFunc"  
#OnAutoItStartRegister "MyTestFunc2"
```

```
Sleep(1000)
```

```
Func MyTestFunc()
```

```
    MsgBox(64, "Start Resultado 2", 'Mensaje de Start desde MyTestFunc()')  
EndFunc
```

```
Func MyTestFunc2()
```

```
    MsgBox(64, "Start Resultado 3", 'Mensaje de Start desde MyTestFunc()')  
EndFunc
```

#NoAutoIt3Execute

Especifica que el actual script compilado no puede correr con los parámetros /AutoIt3ExecuteLine o /AutoIt3ExecuteScript.

```
#NoAutoIt3Execute
```

Parámetros

Ninguno.

Comentarios

El lanzamiento de un script compilado es terminado con el código de salida = -1 si /AutoIt3ExecuteLine o /AutoIt3ExecuteScript es usado.

Relativo

Ninguno.

#RequireAdmin

Especifica que el actual script requiere derechos de administrador.

#RequireAdmin

Parámetros

Ninguno

Comentarios

Esta función fue primeramente creada para proveer un funcionamiento correcto de Autolt para el Control de Cuentas de Usuarios de Windows Vista (UAC) (Sin embargo, también trabaja bajo Windows 2000 y Windows XP). Para más detalles ver [Autolt en Windows Vista](#). Como esta función inicia nuevos procesos, algunas funciones como Consolewrite() no puede ser capturados (SciTe no mostrará nada).

Relativo

[IsAdmin](#)

Ejemplo

```
#RequireAdmin
MsgBox(4096,"Info","Now running with admin rights")
```

Func...Return...EndFunc

Define una Función de Usuario que toma cero o más argumentos y devuelve opcionalmente un resultado.

```
Func NombreFuncion ( [Const] [ByRef] $parametro1, ..., [Const] [ByRef] $parametroN,  
$opcionalpar1 = valor, ...)  
...  
[Return [valor]]
```

EndFunc

Parámetros

Los parámetros son fijados por usted. Luego puede llamar la función desde cualquier otra parte del script.

Comentarios

La palabra clave Const es opcional e indica que el valor del parámetro no cambiará durante la ejecución de la función. Una variable declarada con Const solo puede ser pasada a la función con el parámetro Const.

La palabra clave ByRef indica que el parámetro será tratado como una referencia al objeto original. El comportamiento por defecto es hacer una copia del parámetro a una nueva variable, sin embargo, ByRef vincula la nueva variable con el parámetro original, es decir, se convierte en un alias. ByRef es usado típicamente cuando la función espera un dato voluminoso, como largos arreglos donde el hacer una copia implica un consumo en el rendimiento. Note que no solamente nombres de variables pueden ser pasados como parámetros ByRef, variables temporales no nombradas, como los valores de retorno de una función, también pueden ser pasados como ByRef. Un literal no puede ser pasado como parámetro ByRef.

Si se utiliza las palabras reservadas ByRef y Const conjuntamente como parámetro de función, el orden en que aparecen no tiene importancia, siempre que se coloquen delante del nombre de la variable que se desea modificar.

Arreglos enteros puede ser pasado a funciones (y ser devueltos por ellas) usando simplemente el nombre del arreglo sin los corchetes: "[]". Los Arreglos deberían ser pasados en las funciones-Definidas-Usuario usando la palabra clave ByRef para evitar que se haga una copia de todo el arreglo.

Los parámetros opciones son definidos asignándole un valor predefinido. El valor puede ser una variable global, macro o valor literal. Los parámetros opcionales siempre aparecen al final de la función. Todos los parámetros después del primer parámetro opcional deben ser opcionales. Dentro de la función, el número de parámetros determinados cuando la función es llamada son obtenidos utilizando el macro @NUMPARAMS (ver ejemplo 2).

Use la palabra clave Return para salir de la función. A diferencia de las funciones internas, las funciones-Definidas-Usuario retornan 0 a menos que sea especificado un valor después de Return para ser devuelto por la función.

Note que la declaración de una función no puede aparecer dentro de la declaración de otra función (no pueden ser anidadas).

Relativo

[Dim/Global/Local](#), [#include](#), [Const](#)

Ejemplo

```
Example1()  
Example2()  
  
; example1  
Func Example1()  
; Script de ejemplo con tres funciones de usuario  
; Observe el uso de variables, ByRef, y Return
```

```

Local $foo = 2
Local $bar = 5
msgBox(0,"Hoy es " & today(), "$foo igual " & $foo)
swap($foo, $bar)
msgBox(0,"Después de intercambiar $foo y $bar", "$foo ahora contiene " & $foo)
msgBox(0,"Finalmente", "El mayor de 3 y 4 es " & max(3,4))
EndFunc ;==>Example1

```

```

Func swap(ByRef $a, ByRef $b) ;intercambia el contenido de dos variables
Local $t
$t = $a
$a = $b
$b = $t
EndFunc

```

```

Func today() ;Devuelve la fecha actual en el formato mm/dd/yyyy
    return (@MON & "/" & @MDAY & "/" & @YEAR)
EndFunc

```

```

Func max($x, $y) ;Devuelve el mayor de dos números
If $x > $y Then
    return $x
Else
    return $y
EndIf
EndFunc

```

```

;Final del script de ejemplo 1
; ejemplo2
Func Example2()
; ejemplo utilizando la macro @NumParams
    Test_Numparms(1,2,3,4,5,6,7,8,9,10,11,12,13,14)
EndFunc ;==>Example2

```

```

Func Test_Numparms($v1 = 0, $v2 = 0, $v3 = 0, $v4 = 0, $v5 = 0, $v6 = 0, $v7 = 0, $v8 = 0, $v9
= 0,_
    $v10 = 0, $v11 = 0, $v12 = 0, $v13 = 0, $v14 = 0, $v15 = 0, $v16 = 0, $v17 = 0, $v18 = 0, $v19
= 0)
Local $val
For $i = 1 To @NumParams
    $val &= Eval("v" & $i) & " "
Next
MsgBox(0, "@NumParams example", "@NumParams =" & @NumParams & @CRLF &
@CRLF & $val)
EndFunc

```

;Final del script de ejemplo 2

Dim / Global / Local / Const

Declara una variable, una constante, o crea un arreglo.

Dim [Const] \$variable [= inicialización]

Dim [Const] \$arreglo[subscript 1]...[subscript n] [= inicialización]

Parámetros

const	[opcional] Si esta presente, La palabra clave Const crea una Constante en vez de una variable.
\$variable	El nombre de la variable/constante para ser declarada.
inicialización	El valor con el que se inicializará la variable. Una constante debe tener una inicialización siempre. La inicialización puede ser la llamada a una función.
subscript	El número de elementos que determinan la dimensión del arreglo, indexado por 0 hasta n-1.

Comentarios

Las palabras claves Dim/Local/Global desempeñan funciones semejantes:

1. Declara variables antes de que usted las use (similar a VBScript)
2. Crea un arreglo (vector)

Nota: En Autolt usted puede crear una variable simplemente por asignación de un valor (`$MiVariable = 0`) pero algunas personas prefieren declarar las variables. Si **AutoItSetOption("MustDeclareVars", 1)** es activado, entonces las variables deben ser declaradas.

Usted puede también declarar múltiple variables en una sola línea:

Dim \$a, \$b, \$c

Y también inicializarlas:

Dim \$a = 2, \$b = 10, \$c = 20

Para crear las constantes se sigue de manera similar:

Const \$a = 2, \$b = 10, \$c = 20

Dim Const \$d = 21, \$e = Exp(1)

Local Const \$f = 5, \$g = 7, \$h = -2

Una vez creada, usted no puede cambiar el valor de una constante. También, usted no puede cambiar una variable existente en una constante.

Para inicializar un arreglo, especifique los valores dentro de corchetes ([]), separados por comas. Para múltiples dimensiones, anide las inicializaciones. Usted puede especificar menos

elementos que los declarados, pero no más. También puede utilizar una llamada de función en la inicialización. Si la llamada de la función devuelve un arreglo, entonces el primer elemento del arreglo será tomado.

```
Dim $Array1[12]=[3, 7.5, "string"], $arreglo[5] = [8, 4, 5, 9, 1]
Dim $Grid[2][4]=[[{"Paul", "Jim", "Richard", "Louis"}, [485.44, 160.68, 275.16, 320.00]]
Dim $Test[5] = [3, 1, StringSplit("Abe/Jack/Bobby/Marty", "/"), Cos(0)]
```

La diferencia entre Dim, Local y Global es el entorno en que estas son declaradas:
Dim = Entorno local si el nombre de variable no existe globalmente (en dicho caso la variable rehusa del entorno global!)

Global = Fuerza la creación de la variable a un entorno global

Local = Fuerza la creación de la variable en entorno Local/Function.

Usted debe usar Local o Global, en lugar de Dim, para determinar explícitamente el entorno deseado para variable/constante/arreglo.

Cuando utiliza variables, el entorno local es chequeado primero y después el entorno global.
Cuando creamos un arreglo usted debe limitar a lo sumo 64 dimensiones y/o un total de 16 millones de elementos.

Una característica única de Autolt es la habilidad de copiar arreglos de esta forma:

```
$mycopy = $myarray
```

En este caso \$mycopy será una copia exacta de \$myarray y tendrá sus dimensiones - no es requerido usar Dim para establecer primero las dimensiones.

Si **AutoltSetOption("MustDeclareVars", 1)** esta activo entonces la variable \$mycopy debería ser declarada primero, pero usted no necesita establecer dimensión alguna. Si la variable \$mycopy ya es un arreglo o tiene un valor simple este será borrado antes de que tenga lugar la copia.

Para borrar un arreglo (puede que usted tenga un largo arreglo global y desea liberar memoria), simplemente asigne un valor simple al mismo:

```
$array = 0
```

Esto borrará el arreglo y asignará a la variable el valor 0.

Al declarar el mismo nombre de la variable esta se borrará y será reseteada a los valores actuales. Declarando una variable con un simple valor en el mismo entorno no cambia el valor de la variable.

Si usted declara una variable con el mismo nombre como parámetro, usando **Local** dentro de una función de usuario, un error ocurrirá (no puede haber dos variables locales dentro de una función). **Global** puede ser usado para asignar variables globales dentro de una función, pero si una variable local (o parámetro) tiene el mismo nombre de una variable local, entonces la variable local será usada en vez de la global. Para esto es recomendable asignar nombres distintos a variables locales y globales.

Relativo

[UBound](#), [ReDim](#), [Static](#), [UBound](#), [ReDim](#), [AutoltSetOption](#)

Ejemplo

```
; Ejemplo 1 - Declarando variables
Dim $x, $y = 23, $z
Global $_PI = 3.14159, $RADIUS
Local $_daysWorking = 5

; Ejemplo 2 - Declarando arreglos
Dim $weeklyWorkSchedule[$_daysWorking]
Global $chessBoard[8][8]
Local $mouseCoordinates[2], $windowStats[4]

; Ejemplo 3 - Declarandos variables constantes
Const $x1 = 11, $y1 = 23, $z1 = 55
Global Const $PI = 3.14159, $E = 2.71828
Local Const $daysWorking = 5
```

ContinueCase

En un bloque Select o Switch, aborta el actual Case y continua en el siguiente.

ContinueCase

Parámetros

Ninguno

Comentarios

Normalmente en un bloque Select o Switch, un Case termina cuando la próxima declaración CASE es encontrada. Utilize ContinueCase para ordenar a Autolt que detenga la ejecución del CASE actual e inicie la ejecución del próximo CASE.

Si prueba ejecutar ContinueCase fuera de un bloque Select o Switch se produce un error fatal.

Relativo

[Select...EndSelect](#), [Switch...EndSwitch](#)

Ejemplo

```
$msg = ""  
$szName = InputBox(Default, "Por favor entre una palabra.", "", "M", Default, Default, Default, Default, 10)  
Switch @error  
Case 2  
    $msg = "Intervalo "  
    ContinueCase  
Case 1; Continuando en el previo Case  
    $msg &= "Cacelación"  
Case 0  
    Switch $szName  
    Case "a", "e", "i", "o", "u"  
        $msg = "Vocal"  
    Case "QP"  
        $msg = "Matemáticas"  
    Case "Q" to "QZ"  
        $msg = "Ciencias"  
    Case Else  
        $msg = "Otros"  
    EndSwitch  
Case Else  
    $msg = "Algo salió muy mal."  
EndSwitch  
  
MsgBox(0, Default, $msg)
```

ContinueLoop

Continua un ciclo While/Do/For.

ContinueLoop [nivel]

Parámetros

nivel	[opcional] El nivel del ciclo para reiniciar. Por defecto es 1 (significa el ciclo actual).
-------	--

Comentarios

ContinueLoop continua la ejecución del ciclo testeando la expresión (para declaraciones While, Until o Next).

Un *nivel* negativo o cero no surte efecto.

Aunque ningún programa que use ContinueLoop puede ser reescrito usando la declaración If-ElseIf-EndIf, ContinueLoop puede hacer los Scripts largos más fáciles de entender. Tenga cuidado con ciclos While/Do; usted puede crear ciclos infinitos usando ContinueLoop incorrectamente.

Relativo

[ExitLoop](#), [For](#), [While](#), [Do](#)

Ejemplo

```
;Imprime todos los números de 1 al 10 excepto el número 7  
For $i = 1 to 10  
    If $i = 7 Then ContinueLoop  
    MsgBox(0, "El valor de $i es:", $i)  
Next
```

Default

Palabra reservada usada dentro de llamadas a funciones.

`$var = Default`

Parámetros

Ninguno

Comentarios

Esta palabra reservada no debe ser usada en una expresión de cómputo general. Autolt **no la detecta** por ser de implementación compleja.

Cuando es usado en parámetros a pasar, el comportamiento es específico según la correspondiente documentación de la función.

Para UDFs es el script el responsable de chequear si el parámetro ha sido bien establecido para la palabra reservada Default y que representa el comportamiento que se desea. El parámetro pasado será pasado como palabra reservada Default no como un valor optativo, si es definido.

Relativo

[IsKeyWord](#)

Ejemplo

```
WinMove("[active]","",default,default,200,300) ; solo cambia el tamaño de la ventana activa  
(no la mueve)
```

MyFunc2(Default,Default)

```
Func MyFunc2($Param1 = Default, $Param2 = 'Dos', $Param3 = Default)
```

```
  If $Param1 = Default Then $Param1 = 'Uno'
```

```
  If $Param3 = Default Then $Param3 = 'Tres'
```

```
  MsgBox(0, 'Parámetros', '1 = ' & $Param1 & @LF & _  
        '2 = ' & $Param2 & @LF & _  
        '3 = ' & $Param3)
```

EndFunc

Do...Until

Ciclo basado en una expresión.

Do

declaración

...

Until *<expresión>*

Parámetros

expresión	La declaración entre Do y Until son ejecutado mientras la expresión sea verdadera.
-----------	--

Comentarios

La declaración Do...Until pueden ser anidadas. La expresión es testeada después que el ciclo es ejecutado, de este modo el ciclo será ejecutada uno o más veces.

Relativo

[ContinueLoop](#), [ExitLoop](#)

Ejemplo

```
$i = 0
```

Do

```
  MsgBox(0, "Valor de $i es:", $i)
```

$\$i = \$i + 1$
Until $\$i = 10$

Enum

Enumera constantes.

[entorno] **Enum** [**Step** <ValorGrado>] <constantlist>

Parámetros

entorno	[opcional] El entorno para Enum, puede ser Local, Global, Dim o ninguno. Si es ninguno, es utilizado el entorno para Dim.
ValorGrado	[opcional] Por defecto el grado es adicionando 1. Otros posibles métodos son: *n, +n, -n donde n es el número completo.
constantlist	Una lista de constantes para ser enumeradas.

Comentarios

Por defecto, la primera constante será 0 y el resto será incrementando al valor anterior en 1. Cuando se utiliza el operador multiplicar, la primera constante se le asignará 1 y el resto será incrementada multiplicando el valor anterior por la unidad establecida en (n). Las constantes pueden ser asignadas explícitamente por cualquier declaración válida.

Relativo

Ejemplo

```
Global Enum $E1VAR1, $E1VAR2, $E1VAR3  
MsgBox(4096, "", "Se espera 0: " & $E1VAR1)  
MsgBox(4096, "", "Se espera 1: " & $E1VAR2)  
MsgBox(4096, "", "Se espera 2: " & $E1VAR3)
```

```
Global Enum $E2VAR1 = 10, $E2VAR2, $E2VAR3 = 15  
MsgBox(4096, "", "Se espera 10: " & $E2VAR1)  
MsgBox(4096, "", "Se espera 11: " & $E2VAR2)  
MsgBox(4096, "", "Se espera 15: " & $E2VAR3)
```

```
Global Enum Step *2 $E3VAR1, $E3VAR2, $E3VAR3  
MsgBox(4096, "", "Se espera 1: " & $E3VAR1)  
MsgBox(4096, "", "Se espera 2: " & $E3VAR2)  
MsgBox(4096, "", "Se espera 4: " & $E3VAR3)
```

Exit

Termina la ejecución del script.

Exit [código de retorno]

Parámetros

código de retorno	[opcional] Entero que devuelve el script al finalizar. Este código puede ser usado por Windows o por una variable del DOS %ERRORLEVEL%. Por defecto es 0. Los Scripts normalmente establecen el error a 0 si son ejecutados correctamente; si el error es 1 entonces esto indica típicamente que el script no fue ejecutado correctamente.
-------------------	---

Comentarios

El parámetro, si es incluido, puede ser encerrado entre paréntesis. De esta manera, los siguientes son válidos: **Exit**, **Exit 0**, y **Exit(0)**. Sin embargo, **Exit()** es inválido.

El código puede ser devuelto en una función OnAutoItExitRegister por @EXITCODE.

Relativo

[ExitLoop](#), [OnAutoItExitRegister](#)

Ejemplo

;Primer Ejemplo

Exit

;Segundo Ejemplo

; Termina el script si no existe argumentos de línea de comandos

If \$CmdLine[0] = 0 Then Exit(1)

;Tercero Ejemplo

; Abre un fichero especificado como primer argumento de línea de comandos

\$file = FileOpen(\$CmdLine[1], 0)

; Chequea si el fichero abierto fué leido satisfactoriamente

If \$file = -1 Then Exit(2)

; Si el fichero está vacío termina el script (script satisfactorio)

\$line = FileReadLine(\$file)

If @error = -1 Then Exit

;Coloque el código para procesar el fichero aquí

FileClose(\$file)

Exit ;es opcional como última línea del script

ExitLoop

Termina un ciclo While/Do/For

ExitLoop [nivel]

Parámetros

nivel	[opcional] El número correspondiente del nivel del ciclo que desea terminar. Por defecto es 1 (el ciclo actual).
-------	---

Comentarios

Un *nivel* negativo o cero no surte efecto.

ExitLoop hará terminar un ciclo **While**, **Do** o **For**.

ExitLoop resulta muy útil, de otro modo necesita ejecutar un chequeo de error en ambos Loop-test y el loop-body.

Relativo

[ContinueLoop](#), [Exit](#), [For](#), [Do](#), [While](#)

Ejemplo

```
$sum = 0
While 1 ;uso de un ciclo infinito hasta que ExitLoop sea invocado
    $ans = InputBox("Ejecutando total=" & $sum, _
        " Entre un número positivo. (Un número negativo existe)")
    If $ans < 0 Then ExitLoop
    $sum = $sum + $ans
WEnd
MsgBox(0,"La suma fué", $sum)
```

False / True

Valores boléanos para ser usados en expresiones lógicas.

```
$var = False
$var = True
```

Parámetros

Ninguno

Comentarios

Estas palabras claves no deben ser usados en expresiones en las que AutoIt no detecta su 'uso inadecuado' con resultados impredecibles.

Relativo

Ninguno.

Ejemplo

```
$bool= False  
if NOT $bool = true Then MsgBox(0,"comparación booleana", "OK")
```

For...To...Step...Next

Ciclo basado en una expresión.

For <variable> = <inicio> **To** <parada> [**Step** <incremento>]
declaración

...

Next

Parámetros

variable	variable usada para el contador.
inicio	El valor inicial de la variable.
parada	El valor final de la variable.
incremento	[opcional] El valor numérico (puede ser fraccionario) por el cual el contador es incrementado en cada ciclo. Por defecto es 1.

Comentarios

La variable será creada automáticamente con entorno LOCAL, aun cuando MustDeclareVars sea usado.

Declaraciones For...Next pueden ser anidadas. El ciclo FOR termina cuando el valor de la *variable* excede el valor de *stop*. Si *incremento* o *parada* es una variable, su valor es leído solamente la primera vez que el ciclo es ejecutado.

Un ciclo for es ejecutado cero veces si:

inicio > parada y step ≥ 0, o
inicio < parada y step es < 0

Relativo

[ContinueLoop](#), [ExitLoop](#)

Ejemplo

```
For $i = 5 to 1 Step -1
    MsgBox(0, "Cuenta regresiva!", $i)
Next
MsgBox(0, "", "Gran Suceso!")
```

For...In...Next

Enumera elementos en un objeto de colección en un arreglo.

```
For <$Variable> In <expresión>
    declaración
    ...
Next
```

Parámetros

Variable	Una variable para la cual un elemento es asignado.
expresión	Debe ser una expresión que represente un Objeto, o un arreglo con al menos un elemento.

Comentarios

La variable será creada automáticamente con entorno LOCAL, aun cuando MustDeclareVars sea usado.

Si la expresión es un objeto de colección sin elementos o un arreglo multidimensional, el ciclo será obviado y la variable contendrá una cadena vacía.

Si la expresión no es un Object o un Arreglo, el script se detiene con un error, a menos que un identificador COM Error sea configurado.

Los arreglos en Autolt son de solo lectura cuando utilizamos For...In. Mientras usted puede asignar un valor a la variable dentro del ciclo For...In, este valor no se refleja en el arreglo en sí mismo. Para modificar el contenido de un arreglo durante la enumeración use un ciclo For...To. Declaraciones For...In...Next pueden ser anidadas.

Relativo

[With...EndWith](#)

Ejemplo

```

;Usando un arreglo
Dim $aArray[4]

$aArray[0]="a"
$aArray[1]=0
$aArray[2]=1.3434
$aArray[3]="test"

$string = ""
FOR $element IN $aArray
    $string = $string & $element & @CRLF
NEXT

Msgbox(0,"For..IN Arraytest","Resultado es: " & @CRLF & $string)

```

;Usando un objeto de colección

```

$oShell = ObjCreate("shell.application")
$oShellWindows=$oShell.windows

if Isobj($oShellWindows) then
    $string=""

for $Window in $oShellWindows
    $String = $String & $Window.LocationName & @CRLF
next

msgbox(0,"","Usted tiene las siguientes ventanas abiertas:" & @CRLF & $String)
else

    msgbox(0,"","Usted no ha abierto ninguna ventana.")
endif

```

If...Then

Declaración condicional simple.

If <expresión> **Then** declaración

Parámetros

expresión	Si la expresión es verdadera, la declaración es ejecutada.
-----------	--

Comentarios

Esta versión de la declaración If es usado para ejecutar un enunciado simple sin anteponer nada antes del EndIf.

La *expresión* puede contener operadores boléanos de AND, OR, y NOT así como operadores lógicos <, <=, >, >=, =, ==, y <> agrupados con paréntesis como deseé.

Relativo

If...Else...EndIf, Select...Case...EndSelect, Switch...EndSwitch

Ejemplo

;Termina el script si no existen argumento de línea de comandos

If \$CmdLine[0] = 0 Then Exit

;Alternativa:

If \$CmdLine[0] = 0 Then

Exit

EndIf

If...Else...Else...EndIf

Declaración condicional.

```
If <expresión> Then  
    declaración  
    ...  
    [ElseIf expresión-n Then  
        [elseif declaración ... ]]  
        ...  
    [Else  
        [declaración else]]  
        ...  
EndIf
```

Parámetros

expresión	Si la expresión es verdadera, el primer bloque de código es ejecutado. Si resulta falso lo anterior, entonces la primera condición ElseIf verdadera es ejecutada . De lo contrario, el bloque "Else" es ejecutado.
-----------	--

Comentarios

Las declaraciones IF pueden ser anidadas.

La expresión puede contener operadores boléanos de AND, OR, y NOT así como operadores lógicos <, <=, >, >=, =, ==, y <> agrupados con paréntesis como deseé.

Relativo

[If...Then, Select...Case...EndSelect, Switch...EndSwitch](#)

Ejemplo

```
If $var > 0 Then
    MsgBox(4096, "", "Valor positivo.")
Elseif $var < 0 Then
    MsgBox(4096, "", "Valor negativo.")
Else
    If StringIsXDigit ($var) Then
        MsgBox(4096, "", "Valor debe ser hexadecimal!")
    Else
        MsgBox(4096, "", "El valor debe ser cadena o cero")
    EndIf
EndIf
```

ReDim

Redimensiona un arreglo existente.

ReDim \$arreglo[subscript 1]...[subscript n]

Parámetros

\$arreglo	El nombre de el arreglo para reajustar.
subscript	El número de elementos a crear para la dimensión del arreglo, indexado por 0 a n-1.

Comentarios

La palabra clave ReDim es similar a **Dim**, Excepto que ReDim conserva los valores en el arreglo en lugar de eliminar los valores mientras se reajusta la dimensión del arreglo. El número de dimensiones debe ser el mismo, o el arreglo anterior será ignorado por ReDim. El arreglo debe mantener el entorno (Global o Local) que presentaba antes del reajuste.

Relativo

[Dim, Ubound](#)

Ejemplo

```
; Example Resizing an array
Dim $I, $K, $T, $MSG
Dim $X[4][6], $Y[4][6]

For $I = 0 To 3
    For $K = 0 To 5
        $T = Int(Random(20) + 1) ;Get random numbers between 1 and 20
        $X[$I][$K] = $T
        $Y[$I][$K] = $T
    Next
Next

ReDim $X[3][8]
Dim $Y[3][8]

$MSG = ""
For $I = 0 To UBound($X, 1) - 1
    For $K = 0 To UBound($X, 2) - 1
        If $K > 0 Then $MSG = $MSG & ","
        $MSG = $MSG & $X[$I][$K]
    Next
    $MSG = $MSG & @CR
Next
MsgBox(0, "ReDim Demo", $MSG)

$MSG = ""
For $I = 0 To UBound($Y, 1) - 1
    For $K = 0 To UBound($Y, 2) - 1
        If $K > 0 Then $MSG = $MSG & ","
        $MSG = $MSG & $Y[$I][$K]
    Next
    $MSG = $MSG & @CR
Next
MsgBox(0, "ReDim Demo", $MSG)
```

Select...Case...EndSelect

Declaración condicional

```
Select
    Case <expresión>
        declaración1
```

```

...
[Case
  declaración2
...
]
[Case Else
  declaraciónN
...
]
EndSelect

```

Parámetros

Case <expresión>	Si la expresión es verdadera toda la declaración hasta el próximo Case o declaración EndSelect es ejecutado. Si más de una declaración CASE es verdadera, solamente la primera es ejecutada.
---------------------	--

Comentarios

Las declaraciones Select pueden ser anidadas.

La *expresión* puede contener operadores boléanos de AND, OR, y NOT así como operadores lógicos <, <=, >, >=, =, ==, y <> agrupados con paréntesis como deseé.

Relativo

If...Then, If...Else...EndIf, Switch...EndSwitch, ContinueCase

Ejemplo

```

$var = 0
$var2= ""

```

```

Select
  Case $var = 1
    MsgBox(0, "", "First Case expression was true")
  Case $var2 = "test"
    MsgBox(0, "", "Second Case expression was true")
  Case Else
    MsgBox(0, "", "No preceding case was true!")
EndSelect

```

Static

Declara una variable estática o crea un arreglo estático.

Static [Alcance] \$variable [= inicializador]

Static [Alcance] \$arreglo[dimensión 1]...[dimensión n] [= inicializador]

Parámetros

Alcance	Un modificador opcional, Local o Global que indica donde la variable es visible.
\$variable	El nombre de la variable a declarar estática.
inicializador	El valor que será inicialmente asignado a la variable. El inicializador puede ser una llamada de función que involucre operaciones matemáticas o comparación de cadenas. Este inicializador es solo evaluado la primera vez que la declaración de la variable aparece.
dimensión	El número de elementos que indica la dimensión del arreglo, indexado de 0 a n-1.

Comentarios

La palabra reservada Static puede aparecer en una linea antes del espesificador opcional de alcance , o después. e.j. **Local Static** o **Static Local** ambas formas con aceptables. Si el modificador de alcance **Local** es usado, entonces la variable statics visible y usable solo en la función en qla cual es declarada. Si el modificador de alcance **Global** es usado, entonces la variable static es visible y usable en todas las partes de script; en esta forma, una Global Static no tiene muchas difeencias de una variable Global. Si el modificador de alcance no es usado, entonces la variable static será creada con alcance local; en esta forma, Static es similar a Dim.

La diferencia entre Local y Static es el tiempo de vida de la variable. Las variables Locales son solamente almacenadas mientras la función es llamada y son visibles solo dentro de la función en la cual es declarada; cuando la función retorna, todas estas variables locales son destruidas. Las variables estáticas son asimismo visibles solo en la función donde son declaradas, pero continuan existiendo, es decir, reteniendo el último valor asignado, aún después de que la función halla finalizado. Cuando se acceda a variables, el alcance local es chequeado primero y después el alcance global.

La palabra reservada Static ejecuta una función similar a las palabras reservadas Global/Local/Dim.

1. Todas ellas declaran una variable antes de ser usada.
2. Todas ellas crean un arreglo.

Nota: Las variables Static *deben* ser declaradas usando la palabra reservada **Static** antes de ser usadas, no tiene importancia como AutoItSetOption("MustDeclareVars") es fijado. Las variables estáticas no pueden ser **Const**.

Usted puede también declarar múltiples variables estáticas en una sola línea:

Static \$a, \$b, \$c

E inicializar las variables:

Static \$a = 2, \$b = 10, \$c = 20

Cuando se inicializa una variable static, el valor de inicialización es evaluado y asignado solo la primera vez, cuando la variable es creada. En todos los subsecuentes pasos, el inicializador es ignorado.

Ver Local para más información acerca del uso de arreglos, el cual tiene la misma funcionalizada como en Local, excepto para:

1. La Reinicialización de una variable Static no tendrá efecto
2. El reajuste de tamaño de una arreglo Static es similar a un ReDim.
3. Usted no puede cambiar una variable estática en variable local o global ni viceversa.

Si usted desea reajustar un arrglo, siempre use Static, no Redim.

Relativo

[Local](#), [UBound](#), [ReDim](#), [AutoItSetOption](#)

Ejemplo

; Ejemplos de variables estáticas.

Opt("MustDeclareVars", 1)

Func Test1()
 Static \$STbFirstPass = 1

If \$STbFirstPass Then
 \$STbFirstPass = 0
 ; Ejecutar tareas en el primer tiempo
EndIf
 ; Otras cosas que la función hará
EndFunc ;==>Test1

Func Accumulate(\$State)
 Static \$Values[9]
 Local \$I

If IsNumber(\$State) Then
 Switch \$State
 Case -1
 ; Reseteo
 For \$I = 0 To 8

```

$Values[$I] = 0
Next
Return True
Case -2
Return $Values
Case 0 To UBound($Values) - 1
$Values[$State] += 1
Return $Values[$State]
Case Else
If $State < 0 Then
SetError(1, 0)
Return False
Else
Static $Values[$State + 1]; Reajuste del arreglo para acomodar los nuevos valores.
$Values[$State] = 1
Return 1
EndIf
EndSwitch
Else
SetError(2, 0)
EndIf
EndFunc ;==>Accumulate

```

Global \$I

```

Test1()

For $I = 1 To 99
Accumulate(Random(0, 20, 1))
Next
For $I In Accumulate(-2)
ConsoleWrite($I & ", ")
Next
ConsoleWrite("\n");

```

Test1()

Aviso: Esta característica es experimental. Puede no funcionar, puede contener errores, ser cambiado o removido sin aviso alguno.

No reporte ningún fallo o recomiende nuevas ideas acerca de esta característica.
USAR BAJO SU PROPIO RIESGO.

Switch...Case...EndSwitch

Declaración condicional.

```
Switch <expresión>
  Case <valor> [a <valor>] [,<valor> [a <valor>] ...]
    declaración1
    ...
  [Case <valor> [a <valor>] [,<valor> [a <valor>] ...]
    declaración2
    ...
  ]
  [Case Else
    declaraciónN
    ...
  ]
EndSwitch
```

Parámetros

<expresión>	Una expresión que devuelve un valor. Este valor es entonces comparado con los valores de cada case hasta que se encuentre una coincidencia. Esta expresión es evaluada exactamente una vez dentro de la estructura.
<valor> a<valor>	el case es ejecutado si la expresión esta entre los dos valores.
<valor>	El case es ejecutado si la expresión coincide con el valor.

Comentarios

Si ningún de los casos coincide con el valor que sigue a Switch, entonces la sección Case Else, si esta presente, es ejecutado. Si no coincide ningún cases y la sección Case Else no esta definida, entonces ningún código dentro de la estructura Switch, aparte de la condición, será ejecutado.

Las declaraciones Switch pueden ser anidadas.

Relativo

If...Then, If...Else...EndIf, Select...EndSelect, ContinueCase

Ejemplo

```
Switch @HOUR
Case 6 To 11
  $msg = "Buenas Días"
Case 12 To 17
  $msg = "Buenas Tardes"
Case 18 To 21
  $msg = "Buenas Noches"
Case Else
```

```
$msg = "¿Qué usted esta haciendo arriba?"  
EndSwitch
```

```
MsgBox(0, Default, $msg)
```

With...EndWith

Usado para reducir referencias largas a variables de tipo objeto.

```
With <expresión>  
    declaración  
    ...  
EndWith
```

Parámetros

expresión	Debe ser una expresión de tipo objeto.
-----------	--

Comentarios

Las declaraciones With...EndWith NO pueden ser anidadas.

Relativo

For...In...Next

Ejemplo

```
$oExcel = ObjCreate("Excel.Application")  
$oExcel.visible =1  
$oExcel.workbooks.add
```

```
With $oExcel.activesheet  
    .cells(2,2).value = 1  
    .range("A1:B2").clear  
EndWith
```

```
$oExcel.quit
```

While...WEnd

Ciclo basado en una expresión.

While <expresión>

 declaración

...

WEnd

Parámetros

expresión	Si la expresión es verdadera la toda la declaración hasta la declaración WEnd es ejecutado. El ciclo continua indefinidamente mientras la expresión sea verdadera.
-----------	--

Comentarios

Las declaraciones While...WEnd pueden ser anidadas.

La expresión es testeada antes que el ciclo sea ejecutado de esta manera el ciclo es ejecutado cero o más veces.

Para crear un ciclo infinito, usted puede usar un número diferente de cero en *expresión*.

Relativo

[ContinueLoop](#), [ExitLoop](#)

Ejemplo

$\$i = 0$

While $\$i \leq 10$

MsgBox(0, "Value de $\$i$ es:", $\$i$)

$\$i = \$i + 1$

WEnd

Referencia de Macros

Debajo hay una lista en orden alfabético de todos los marcos disponibles en AutoIt.

Macro	Descripción
@AppDataCommonDir	Ruta a la carpeta común de Datos de aplicación.
@AppDataDir	Ruta de la carpeta de Datos de Aplicación del usuario actual.
@AutoItExe	La dirección completa y nombre de archivo del archivo ejecutable AutoIt actualmente en ejecución. Para archivos compilados es la ruta completa del script compilado.
@AutoItPID	PID del proceso ejecutando el script.
@AutoItVersion	Número de Versión de AutoIt tal como 3.2.1.0
@AutoItX64	Devuelve 1 si el script está ejecutándose bajo la versión nativa de x64 de AutoIt.
@COM_EventObj	Objeto del evento COM está siendo notificado. Válido solamente en la función de eventos COM.
@CommonFilesDir	ruta a la carpeta de Archivos Comunes (Common Files)
@Compiled	Devuelve 1 si el script es ejecutable compilado; de otra manera, devuelve 0.
@ComputerName	Nombre de la Computadora en la red.
@ComSpec	valor de %comspec%, el interpretador secundario especificado de comandos; primordialmente para usos de línea de comando, ejemplo <i>Run(@ComSpec & "/k help / more")</i>
@CPUArch	Devuelve "X86" cuando la CPU es una CPU de 32-bit y "X64" cuando la CPU es de 64-bit.
@CR	Retorno de Carro, Chr(13); algunas veces es usado para interrupción de línea.
@CRLF	= @CR & @LF; ocasionalmente usado para interrupción de líneas.
@DesktopCommonDir	ruta a la carpeta de Escritorio (Desktop)
@DesktopDir	ruta a la carpeta del Escritorio del usuario
@DesktopHeight	Altura de la pantalla del escritorio en píxeles. (resolución vertical)
@DesktopWidth	Ancho de la pantalla del escritorio en píxeles. (resolución horizontal)
@DesktopDepth	Rango de la pantalla del escritorio en bits por píxel.

@DesktopRefresh	Tasa de refresco de la pantalla del escritorio en hercios.
@DocumentsCommonDir	ruta de la carpeta a Documentos
@error	Bandera de estado de error. Ver función SetError.
@exitCode	Código de Salida establecido por el enunciado Exit.
@exitMethod	Método de salida. Ver la Función OnAutoItExitRegister().
@extended	Valor devuelto por la función Extended - usado en ciertas funciones tal como StringReplace.
@FavoritesCommonDir	ruta a Favoritos
@FavoritesDir	ruta a la carpeta de Favoritos del usuario actual
@GUI_CtrlId	Identificador del último control pulsado. Válido solamente en una función de evento. Ver la función GUICtrlSetOnEvent.
@GUI_CtrlHandle	Identificador del último handle pulsado. Válido solamente en una función de evento. Ver la función GUICtrlSetOnEvent.
@GUI_DragId	Identificador del último control arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_DragFile	Nombre del archivo que está siendo arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_DropId	Identificador del control arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_WinHandle	El handle de la última ventana GUI pulsada. Válido solamente en Función de evento. Ver la función GUICtrlSetOnEvent.
@HomeDrive	Letra de la unidad conteniendo al directorio de inicio del usuario.
@HomePath	Parte del Directorio de Inicio del usuario actual. Para obtener la dirección completa, usarlo en conjunto con @HomeDrive.
@HomeShare	Servidor y nombre de recurso compartido del directorio inicio del usuario actual.
@HOUR	Valor de Horas del reloj en formato de 24-horas. Rango es 00 a 23
@HotKeyPressed	Último hotkey presionado. Ver la función HotKeySet.
@IPAddress1	Dirección IP del primer adaptador de red. Tiende a devolver 127.0.0.1 en algunas computadoras.
@IPAddress2	Dirección IP del segundo adaptador de red. Devuelve 0.0.0.0 si no es aplicable.

@IPAddress3	Dirección IP del tercer adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@IPAddress4	Dirección IP del cuarto adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@KBLLayout	Devuelve código de acuerdo al esquema del Teclado. Ver Apéndice para posibles valores.
@LF	Avance de línea, Chr(10); típicamente usado para saltos de línea.
@LogonDNSDomain	Inicio de sesión de dominio de DNS.
@LogonDomain	Inicio de sesión de dominio.
@LogonServer	Inicio de sesión de servidor.
@MDAY	Día del mes actual. Rango es de 01 a 31
@MIN	Valor de los minutos del reloj. Rango es de 00 a 59
@MON	Mes en curso. Rango es de 01 a 12
@MSEC	Valores de Milisegundos de reloj. Rango entre 00 y 999
@MUILang	Retorna el código multilenguaje si es disponible (En Vista funciona por defecto). Ver Apéndice para posibles valores.
@MyDocumentsDir	ruta de la carpeta Mis Documentos
@NumParams	Número de parámetros utilizados para llamar a los usuarios funciones
@OSArch	Devuelve un valor de los siguientes: "X86", "IA64", "X64" - Esto es el tipo de arquitectura del sistema operativo actual en ejecución.
@OSBuild	Devuelve el número de build del Sistema Operativo. Por ejemplo, Windows 2003 Server devuelve 3790
@OSLang	Devuelve código de acuerdo al lenguaje del Sistema Operativo. Ver Apéndice para valores posibles.
@OSServicePack	Información de Service pack en la forma de "Service Pack 3"
@OSTYPE	Devuelve "WIN32_NT" para NT/2000/XP/2003/Vista/2008/Win7/2008R2.
@OSVersion	Devuelve uno de los siguientes: "WIN_2008R2", "WIN_7", "WIN_2008", "WIN_VISTA", "WIN_2003", "WIN_XP", "WIN_XPe", "WIN_2000".
@ProcessorArch	Devuelve un valor de los siguientes: "WIN_2008", "WIN_VISTA", "WIN_2003", "WIN_XP", "WIN_2000"
@ProgramFilesDir	ruta a la carpeta de Archivos de Programa
@ProgramsCommonDir	ruta a la carpeta del Menú de Inicio

@ProgramsDir	ruta a la carpeta de Programas del usuario actual (carpeta de Menú de Inicio)
@ScriptDir	Directorio contenido el script ejecutándose. (El resultado no contiene una pleca al final)
@ScriptFullPath	Equivalente a @ScriptDir & "\" & @ScriptName
@ScriptLineNumber	Número de línea que está siendo ejecutada en el script actual. Útil para enunciados de depuración especialmente cuando una función es llamada para evaluarse. (No es significante en un script compilado)
@ScriptName	Nombre largo del archivo script en ejecución.
@SEC	Valor en segundos de la hora. Rango es 00 a 59
@StartMenuCommonDir	ruta de acceso para el menú Inicio común
@StartMenuDir	ruta de acceso para el menú Inicio del usuario actual
@StartupCommonDir	ruta a la carpeta de Inicio
@StartupDir	carpeta del inicio del usuario actual
@SW_DISABLE	Desactiva la ventana.
@SW_ENABLE	Activa la ventana.
@SW_HIDE	Oculta la ventana y activa otra.
@SW_LOCK	Bloquea la ventana evitando así pintar sobre ella.
@SW_MAXIMIZE	Maximiza la ventana especificada.
@SW_MINIMIZE	Minimiza la ventana especificada y activa la próxima al frente según el orden en Z.
@SW_RESTORE	Activa y muestra la ventana. Si la ventana está minimizada o maximizada, el sistema restaura a su posición y tamaño original. Una aplicación debería especificar esta bandera cuando se restaura una ventana minimizada.
@SW_SHOW	Activa la ventana y muestra en su posición y tamaño actual.
@SW_SHOWDEFAULT	Establece el estado de muestra de una ventana basado en el valor SW_ especificado por el programa que inició la aplicación.
@SW_SHOWMAXIMIZED	Activa la ventana y la muestra como ventana maximizada.
@SW_SHOWMINIMIZED	Activa la ventana y la muestra como ventana minimizada.
@SW_SHOWMINNOACTIVE	Muestra la ventana como ventana minimizada. Este valor es similar a @SW_SHOWMINIMIZED, excepto que la ventana no es activada.
@SW_SHOWNA	Muestra la ventana en su tamaño actual y posición. Este valor es similar a @SW_SHOW, excepto que la ventana no es activada.

@SW_SHOWNOACTIVATE	Muestra una ventana en su más reciente posición y tamaño. Este valor es similar a @SW_SHOWNORMAL, excepto que la ventana no es activada.
@SW_SHOWNORMAL	Activa y muestra una ventana. Si la ventana está minimizada o maximizada, el sistema restaura su posición y tamaño original. Una aplicación debería especificar esta bandera cuando está mostrando la ventana por primera vez.
@SW_UNLOCK	Desbloquea una ventana bloqueada para re muestreo.
@SystemDir	ruta a la carpeta de Windows' System (o el System32)
@TAB	carácter Tab, Chr(9)
@TempDir	Ruta a la carpeta de archivos temporales.
@TRAY_ID	Identificador del último ítem pulsado durante una acción en un TraySet(Item)OnEvent.
@TrayIconFlashing	Devuelve 1 si el ícono de bandeja está parpadeando; de otra manera, devuelve 0.
@TrayIconVisible	Devuelve 1 si el ícono de bandeja está visible; de otra manera, devuelve 0.
@UserProfileDir	Ruta a la carpeta del perfil del usuario actual.
@UserName	ID del usuario actual que ha iniciado sesión.
@WDAY	Numérico el día de la semana. La escala es de 1 a 7, que corresponde de domingo a sábado.
@WindowsDir	ruta a la carpeta de Windows
@WorkingDir	Directorio actual de trabajo activo. (Los resultados no contiene una barra invertida)
@YDAY	Día actual del año. El rango es de 1 a 366 (o 001 a 365 si no es un año bisiesto)
@YEAR	Año actual en cuatro-dígitos

Macros relativos a Autolt

Deabajo hay una lista con los macros relativos a Autolt.

Macro	Description
@Compiled	Devuelve 1 si el script es ejecutable compilado; de otra manera, devuelve 0.
@error	Bandera de estado de error. Ver función SetError.

@exitCode	Código de Salida establecido por el enunciado Exit.
@exitMethod	Método de salida. Ver la Función OnAutoItExitRegister()
@extended	Valor devuelto por la función Extended - usado en ciertas funciones tal como StringReplace.
@NumParams	Número de parámetros utilizados para llamar a los usuarios funciones
@ScriptName	Nombre largo del archivo script en ejecución.
@ScriptDir	Directorio contenido el script ejecutándose. (El resultado no contiene una pleca al final)
@ScriptFullPath	Equivalente a @ScriptDir & "" & @ScriptName
@ScriptLineNumber	Número de línea que está siendo ejecutada en el script actual. Útil para enunciados de depuración especialmente cuando una función es llamada para evaluarse. (No es significante en un script compilado)
@WorkingDir	Directorio actual de trabajo activo. (Los resultados no contiene una barra invertida)
@AutoItExe	La dirección completa y nombre de archivo del archivo ejecutable AutoIt actualmente en ejecución. Para archivos compilados es la ruta completa del script compilado.
@AutoItPID	PID del proceso ejecutando el script.
@AutoItVersion	Número de Versión de AutoIt tal como 3.2.1.0
@AutoItX64	Devuelve 1 si el script está ejecutándose bajo la versión nativa de x64 de AutoIt.
@COM_EventObj	Objeto del evento COM está siendo notificado. Válido solamente en la función de eventos COM.
@GUI_CtrlId	Identificador del último control pulsado. Válido solamente en una función de evento. Ver la función GUICtrlSetOnEvent.
@GUI_CtrlHandle	Identificador del último handle pulsado. Válido solamente en una función de evento. Ver la función GUICtrlSetOnEvent.
@GUI_DragID	Identificador del último control arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_DragFile	Nombre del archivo que está siendo arrastrado. Válido solamente en una función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_DropID	Identificador del control arrastrado. Válido solamente en una

	función de Evento de Arrastre (Drop Event). Ver la función GUISetOnEvent.
@GUI_WinHandle	Last click GUI Window handle. Only valid in an event Function. Ver la Función GUICtrlSetOnEvent.
@HotKeyPressed	El handle de la última ventana GUI pulsada. Válido solamente en Función de evento. Ver la función GUICtrlSetOnEvent.

Para usar con las funciones [WinsetState](#), [Run](#), [RunWait](#), [FileCreateShortcut](#) y [FileGetShortcut](#):

@SW_DISABLE	Desactiva la ventana.
@SW_ENABLE	Activa la ventana.
@SW_HIDE	Oculta la ventana y activa otra.
@SW_LOCK	Bloquea la ventana evitando así pintar sobre ella.
@SW_MAXIMIZE	Maximiza la ventana especificada.
@SW_MINIMIZE	Minimiza la ventana especificada y activa la próxima al frente según el orden en Z.
@SW_RESTORE	Activa y muestra la ventana. Si la ventana está minimizada o maximizada, el sistema restaura a su posición y tamaño original. Una aplicación debería especificar esta bandera cuando se restaura una ventana minimizada.
@SW_SHOW	Activa la ventana y muestra en su posición y tamaño actual.
@SW_SHOWDEFAULT	Establece el estado de muestra de una ventana basado en el valor SW_ especificado por el programa que inició la aplicación.
@SW_SHOWMAXIMIZED	Activa la ventana y la muestra como ventana maximizada.
@SW_SHOWMINIMIZED	Activa la ventana y la muestra como ventana minimizada.
@SW_SHOWMINNOACTIVE	Muestra la ventana como ventana minimizada. Este valor es similar a @SW_SHOWMINIMIZED, excepto que la ventana no es activada.
@SW_SHOWNA	Muestra la ventana en su tamaño actual y posición. Este valor es similar a @SW_SHOW, excepto que la ventana no es activada.
@SW_SHOWNOACTIVATE	Muestra una ventana en su más reciente posición y tamaño. Este valor es similar a @SW_SHOWNORMAL, excepto que la ventana no es activada.
@SW_SHOWNORMAL	Activa y muestra una ventana. Si la ventana está minimizada o maximizada, el sistema restaura su posición y tamaño original. Una aplicación debería especificar esta bandera cuando está mostrando la ventana por primera vez.
@SW_UNLOCK	Desbloquea una ventana bloqueada para re muestreo.

@TRAY_ID	Identificador del último Item pulsado durante una acción en un TraySet(Item)OnEvent.
@TrayIconFlashing	Devuelve 1 si el ícono de bandeja está parpadeando; de otra manera, devuelve 0.
@TrayIconVisible	Devuelve 1 si el ícono de bandeja está visible; de otra manera, devuelve 0.
@CR	Retorno de Carro, Chr(13); algunas veces es usado para interrupción de línea.
@LF	Avance de línea, Chr(10); típicamente usado para saltos de línea.
@CRLF	= @CR & @LF; ocasionalmente usado para interrupción de líneas.
@TAB	carácter Tab, Chr(9)

Macros Relativos a Directorios

Debajo hay una lista con los macros de Directorios. Note que las rutas NO contienen la barra al final.

Macro	Descripción
Macros para los datos de "Todos los usuarios".	
@AppDataCommonDir	ruta a la carpeta común de Datos de aplicación
@DesktopCommonDir	ruta a la carpeta de Escritorio (Desktop)
@DocumentsCommonDir	ruta de la carpeta a Documentos
@FavoritesCommonDir	ruta a Favoritos
@ProgramsCommonDir	ruta a la carpeta del Menú de Inicio
@StartMenuCommonDir	ruta de acceso para el menú Inicio común
@StartupCommonDir	ruta a la carpeta de Inicio
Macros para los datos del usuario actual.	
@AppDataDir	ruta de la carpeta de Datos de Aplicación del usuario actual
@DesktopDir	ruta a la carpeta del Escritorio del usuario
@MyDocumentsDir	ruta de la carpeta Mis Documentos
@FavoritesDir	ruta a la carpeta de Favoritos del usuario actual
@ProgramsDir	ruta a la carpeta de Programas del usuario actual (carpeta de Menú de Inicio)

@StartMenuDir	ruta de acceso para el menú Inicio del usuario actual
@StartupDir	carpeta del inicio del usuario actual
@UserProfileDir	Ruta a la carpeta del perfil del usuario actual.
Otros macros para el sistema:	
@HomeDrive	Letra de la unidad conteniendo al directorio de inicio del usuario.
@HomePath	Parte del Directorio de Inicio del usuario actual. Para obtener la dirección completa, usarlo en conjunto con @HomeDrive.
@HomeShare	Servidor y nombre de recurso compartido del directorio inicio del usuario actual.
@LogonDNSDomain	Inicio de sesión de dominio de DNS.
@LogonDomain	Inicio de sesión de dominio.
@LogonServer	Inicio de sesión de servidor.
@ProgramFilesDir	ruta a la carpeta de Archivos de Programa
@CommonFilesDir	ruta a la carpeta de Archivos Comunes (Common Files)
@WindowsDir	ruta a la carpeta de Windows
@SystemDir	ruta a la carpeta de Windows' System (o el System32)
@TempDir	Ruta a la carpeta de archivos temporales.
@ComSpec	valor de %comspec%, el interpretador secundario especificado de comandos; primordialmente para usos de línea de comando, ejemplo <i>Run(@ComSpec & "/k help more")</i>

Macros Relativos a Información del Sistema

Deabajo hay una lista de los macros de información del Sistema.

Macro	Descripción
@CPUArch	Devuelve "X86" cuando la CPU de de 32-bit y "X64" cuando la CPU es de 64-bit.
@KBLLayout	Devuelve código de acuerdo al esquema del Teclado. Ver Apéndice para posibles valores.
@MUILang	Devuelve el código indicando Multi Lenguaje si es disponible (en Vista trabaja bien por defecto). Ver Apéndice para valores posibles.
@OSArch	Devuelve un valor de los siguientes: "X86", "IA64", "X64" - Esto es el tipo de arquitectura del Sistema Operativo actualmente en ejecución.

@OSLang	Devuelve código de acuerdo al lenguaje del Sistema Operativo. Ver Apéndice para valores posibles.
@OSTYPE	Devuelve "WIN32_NT" for NT/2000/XP/2003/Vista/2008/Win7/2008R2.
@OSVersion	Devuelve un valor de los siguientes: "WIN_2008R2", "WIN_7", "WIN_2008", "WIN_VISTA", "WIN_2003", "WIN_XP", "WIN_XPe", "WIN_2000".
@OSBuild	Devuelve el número de build del Sistema Operativo. Por ejemplo, Windows 2003 Server devuelve 3790
@OSServicePack	Información de Service pack en la forma de "Service Pack 3" o, para Windows 95, devolverá "B"
@ComputerName	Nombre de la Computadora en la red.
@UserName	ID del usuario actual que ha iniciado sesión.
@IPAddress1	Dirección IP del primer adaptador de red. Tiende a devolver 127.0.0.1 en algunas computadoras.
@IPAddress2	Dirección IP del segundo adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@IPAddress3	Dirección IP del tercer adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@IPAddress4	Dirección IP del cuarto adaptador de red. Devuelve 0.0.0.0 si no es aplicable.
@DesktopHeight	Altura de la pantalla del escritorio en píxeles. (resolución vertical)
@DesktopWidth	Ancho de la pantalla del escritorio en píxeles. (resolución horizontal)
@DesktopDepth	Rango de la pantalla del escritorio en bits por píxel.
@DesktopRefresh	Tasa de refresco de la pantalla del escritorio en hercios.

Marcos Relativos a Fecha y Hora

Debajo hay una lista de macros relacionados con Tiempo y Fecha macros. Note que la mayoría de los valores devueltos, su longitud es de dos dígitos.

Macro	Descripción
@MSEC	Valores milisegundos de reloj. Rango entre 00 y 999
@SEC	Valor en segundos de la hora. Rango es 00 a 59

@MIN	Valor de los minutos del reloj. Rango es de 00 a 59
@HOUR	Valor de Horas del reloj en formato de 24-horas. Rango es 00 a 23
@MDAY	Día del mes actual. Rango es de 01 a 31
@MON	Mes en curso. Rango es de 01 a 12
@YEAR	Año actual en cuatro-dígitos
@WDAY	Numérico el día de la semana. La escala es de 1 a 7, que corresponde de domingo a sábado.
@YDAY	Día actual del año. El rango es de 1 a 366 (o de 001 a 365 si no es un año bisiesto)

REFERENCIA DE FUNCIONES

Arriba ya tenéis la lista de las principales funciones de autoit... no la vamos a escribir otra vez, si bien os pasamos a clasificarlas y explicarlas

Notas de Funciones

Muchas funciones contienen parámetros opcionales que pueden ser omitidos. Si desea especificar un parámetro opcional, todos los parámetros anteriores deben ser especificados! Por ejemplo, considere [Run](#) ("nombredearchivo", ["dirDeTrabajo"] [, bandera]). Si se desea especificar la *bandera*, se **debe** especificar *directorioDeTrabajo*.

Muchas funciones Win___ contienen un parámetro opcional llamado "texto". Este parámetro está destinado para ayudar a diferenciar entre ventanas que tienen títulos idénticos.

Algunas funciones indican éxito/fallo como **valor devuelto**; otros se indican fijando en la **flag de @error**. Ambos tienen las siguientes significación...
@error = 0 ; cuando es igual a cero, es siempre exitoso
Return = varía, pero típicamente valores no-cero exitosos permiten leer fácilmente el código...

```
If algunaFuncUsuario() then ; ...función trabajó
If Not someUserFunc() then ; ...función falló
$x = FileReadLine("C:\someFile.txt")
If @error = -1 Then ;fin-de-archivo fué lanzado
```

Si una función puede establecer el flag de @error, se debería revisar siempre antes de usar un valor devuelto - si @error indica un error entonces el valor devuelto de la función es generalmente indefinido...

@error es siempre establecido a 0 cuando entra en una función.

Cuando la documentación establece que el valor devuelto es = ninguno, AutoIt siempre devuelve un valor para evitar errores. Valor de 1 es usualmente devuelto, pero no debería depender de este valor devuelto.

Cuando un parámetro opcional necesita ser definido y es precedido por uno o más parámetrosopcionales, los parámetros anteriores deben ser dados. Esto tal vez sea "" para un parámetro de cadena de texto y -1 para otros tipos. Algunas funciones como StringInStr o StringReplace requieren 0. Ver la descripción correspondiente del parámetro opcional en cada caso.

Referencia de Administración de entorno

A continuación se muestra una lista completa de las funciones de Administración de entorno disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
ClipGet	Recupera texto del portapapeles.
ClipPut	Escribe texto en el portapapeles.
EnvGet	Recupera una variable de entorno.
EnvSet	Escribe en una variable de entorno.
EnvUpdate	Refresca el entorno del SO.
MemGetStats	Devuelve información relativa a la memoria.

ClipGet

Devuelve el texto del portapapeles.

ClipGet

Parámetros

Ninguno.

Valor de retorno

Con Éxito: Devuelve una cadena conteniendo el texto en el portapapeles.

Al Fallar: Establece oAutolt.error a 1 si el portapapeles esta vacío o no contiene texto.

Comentarios

Ninguno.

Relativo

[ClipPut](#)

Ejemplo

```
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
texto = oAutoIt.ClipGet()
WScript.Echo "Clipboard contains:" & texto
```

ClipPut

Escribe texto al portapapeles.

ClipPut "valor"

Parámetros

valor	El texto para ser escrito al portapapeles.
-------	--

Valor de retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0.

Comentarios

Cualquier contenido en el portapapeles es sobrescrito.

Relativo

[ClipGet](#)

Ejemplo

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

EnvGet

Recupera una variable de entorno.

EnvGet ("Variable")

Parámetros

Variable	Nombre de la variable de entorno a acceder tales como "TEMP" o "PATH".
----------	--

Valor de Retorno

Devuelve la variable solicitada (o una cadena vacía si la variable no existe).

Comentarios

Ninguno

Relativo

[EnvSet](#), [EnvUpdate](#)

Ejemplo

```
$var = EnvGet("PATH")
MsgBox(4096, "Path de la variable es:", $var)
```

EnvSet

Escribe en una variable de entorno.

EnvSet ("Variable" [, "valor"])

Parámetros

Variable	Nombre de la variable de entorno para acceder.
valor	[opcional] Valor a fijar en la variable de entorno. Si un valor no es usado la variable será eliminada.

Valor de Retorno

Con Éxitos Devuelve un valor diferente de 0.

Al Fallar Devuelve 0.

Comentarios

Una variable de entorno establecida de esta forma solamente puede ser accedida por programas como AutoIt (Run, RunWait). Una vez que AutoIt finaliza, la variable deja de existir.

Relativo

[EnvGet](#), [EnvUpdate](#)

Ejemplo

EnvSet("MYENV", "esto es un test")

EnvUpdate

Refresca el entorno del SO.

EnvUpdate()

Parámetros

Ninguno

Valor de Retorno

Con Éxitos Ninguno

Al Fallar Establece @error a 1.

Comentarios

El efecto es similar a reiniciar el sistema. Por ejemplo, al cambiar la ruta de entorno %ruta% este cambio no tendrá efecto hasta que no se invoque EnvUpdate (o Reinicie/apague el sistema operativo).

Relativo

[EnvGet](#), [EnvSet](#)

Ejemplo

EnvUpdate()

MemGetStats

Devuelve información relativa a la memoria.

MemGetStats()

Parámetros

Ninguno

Valor de Retorno

Devuelve un arreglo de siete elementos contenido información de la memoria:

*\$arreglo[0] = Carga de memoria (Porcentaje de memoria en uso)
\$arreglo[1] = RAM física total
\$arreglo[2] = RAM física disponible
\$arreglo[3] = Archivo de paginación total
\$arreglo[4] = Archivo de paginación disponible
\$arreglo[5] = Virtual total
\$arreglo[6] = Virtual disponible*

Todas las unidades de memoria se expresan en **kilobytes**.

Comentarios

Ninguno

Relativo

Ninguno.

Ejemplo

```
$mem = MemGetStats()  
  
MsgBox(0, "RAM física total (KB):", $mem[1])
```

Referencia de Administración de fichero, Directorio y Disco

A continuación se muestra una lista completa de las funciones de Administración de fichero, Directorio y Disco disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
ConsoleRead	Lee datos desde la secuencia STDIN del proceso de script de AutoIt.
ConsoleWrite	Escribe datos en la secuencia de STDOUT. Algunos editores de texto pueden leer dicha secuencia como otros programas.
ConsoleWriteError	Escribe datos en la secuencia de STDERR. Algunos editores de texto pueden leer dicha secuencia como otros programas.
DirCopy	Copia un directorio conjuntamente con sus subdirectorios y archivos (Similar a xcopy).
DirCreate	Crea un directorio/carpeta.
DirGetSize	Devuelve el tamaño en bytes de un directorio dado.
DirMove	Mueve un directorio y todos sus subdirectorios y ficheros.
DirRemove	Borra un directorio/carpeta.
DriveGetDrive	Devuelve un arreglo devolviendo las unidades enumeradas.
DriveGetFileSystem	Devuelve el Tipo de Sistema de Archivo de una unidad.
DriveGetLabel	Devuelve la Etiqueta de Volumen de una unidad, si lo posee.
DriveGetSerial	Devuelve el Número de Serie de una unidad.
DriveGetType	Devuelve tipo de unidad.

<u>DriveMapAdd</u>	Mapea una unidad de red.
<u>DriveMapDel</u>	Desconecta una unidad de red.
<u>DriveMapGet</u>	Recupera información detallada de una unidad mapeada.
<u>DriveSetLabel</u>	Establece la Etiqueta de Volumen de una unidad.
<u>DriveSpaceFree</u>	Devuelve el espacio libre en disco(en Megabytes)de una ruta en Megabytes.
<u>DriveSpaceTotal</u>	Devuelve el espacio total libre en disco de una ruta en Megabytes.
<u>DriveStatus</u>	Devuelve el estado de una unidad como una cadena.
<u>FileChangeDir</u>	Cambia el directorio actual de trabajo.
<u>FileClose</u>	Cierra un fichero de texto previamente abierto.
<u>FileCopy</u>	Copia uno o más ficheros.
<u>FileCreateNTFSLink</u>	Crea un hardlink NTFS para un archivo o directorio.
<u>FileCreateShortcut</u>	Crea un acceso directo (.lnk) a un fichero.
<u>FileDelete</u>	Elimina uno o más ficheros.
<u>FileExists</u>	Chequea si un archivo o directorio existe.
<u>FileFindFirstFile</u>	Devuelve un identificador de búsqueda de acuerdo con una cadena de búsqueda.
<u>FileFindNextFile</u>	Devuelve un nombre de archivo de acuerdo con una invocación previa a FileFindFirstFile.
<u>FileFlush</u>	Limpia el buffer de una fichero en disco.
<u>FileGetAttrib</u>	Devuelve una cadena codificada conteniendo los atributos de un fichero.

<u>FileGetEncoding</u>	Determina la codificación de texto utilizada en un fichero.
<u>FileGetLongName</u>	Devuelve el nombre completo (dirección+nombre) de una ruta dada.
<u>FileGetPos</u>	Devuelve la posición actual en un fichero
<u>FileGetShortcut</u>	Devuelve detalles de un acceso directo.
<u>FileGetShortName</u>	Devuelve el nombre corto 8.3 de una dirección+nombre de archivo dado.
<u>FileGetSize</u>	Devuelve el tamaño de un archivo en bytes.
<u>FileGetTime</u>	Devuelve información del tiempo y la fecha para un fichero.
<u>FileGetVersion</u>	Devuelve información de la versión del "fichero".
<u>FileInstall</u>	Incluye e instala un archivo con el archivo compilado.
<u>FileMove</u>	Mueve uno o más directorios.
<u>FileOpen</u>	Abre un archivo de texto para lectura o escritura.
<u>FileOpenDialog</u>	Inicia un cuadro de diálogo de Abrir Archivo.
<u>FileRead</u>	Lee un número de caracteres desde un fichero previamente abierto.
<u>FileReadLine</u>	Lee una línea de texto desde un fichero previamente abierto.
<u>FileRecycle</u>	Envía un archivo o directorio a la papelera de reciclaje (recycle bin).
<u>FileRecycleEmpty</u>	Vacía la papelera de reciclaje.
<u>FileSaveDialog</u>	Inicializa un diálogo de Salvar Fichero.
<u>FileSelectFolder</u>	Inicializa un diálogo de selección de carpeta.
<u>FileSetAttrib</u>	Establece los atributos de uno o más ficheros.

<u>FileSetPos</u>	Establece la posición actual en el fichero.
<u>FileSetTime</u>	Establece la marca de tiempo de uno o más ficheros
<u>FileWrite</u>	Agrega un texto/dato al final de un fichero previamente abierto.
<u>FileWriteLine</u>	Agrega una línea de texto al final de un fichero de texto previamente abierto.
<u>IniDelete</u>	Borra un valor de un fichero .ini con formato estándar.
<u>IniRead</u>	Lee un valor desde un fichero .ini con formato estándar.
<u>IniReadSection</u>	Lee todos los pares de valores/claves de una sección en un fichero .ini con formato estándar.
<u>IniReadSectionNames</u>	Lee todas las secciones en un fichero .ini con formato estándar.
<u>IniRenameSection</u>	Renombra una sección en un fichero .ini con formato estándar.
<u>IniWrite</u>	Escribe un valor para un fichero .ini con formato estándar.
<u>IniWriteSection</u>	Escribe una sección para un fichero .ini con formato estándar.

ConsoleRead

Lee datos desde la secuencia STDIN del proceso de script de AutoIt.

ConsoleRead ([peek = false[, binario = false]])

Parámetros

peek	[optional] Si es true la función no remueve los caracteres leídos desde la secuencia.
binario	[optional] Si es true la función lee el dato como binario en vez de texto (por defecto es texto).

Valor de Retorno

- Con Éxitos Devuelve el dato leído. @extended contiene el número de bytes leídos.
- Al Fallar Establece @error a no-cero si EOF es alcanzado, STDIN no está conectado a otro proceso o error.

Comentarios

ConsoleRead lee desde la secuencia de la consola estándar de los procesos de script de AutoIt, que es normalmente usado como consola de aplicación para leer las entradas de los procesos. ConsoleRead no bloquea, esta devolverá inmediatamente. Para obtener todos los datos, debe ser invocado en un ciclo.

Examinando la secuencia de entrada no se borran los datos del buffer, sin embargo, este no devuelve el dato disponible como normal.

Por defecto, el dato es devuelto en formato texto. Utilizando operaciones binarias, el dato puede ser devuelto en formato binario.

Relativo

[ConsoleWrite](#), [ConsoleWriteError](#), [Run](#)

Ejemplo

; Compilar este script a "ConsoleRead.exe".
; Abrir el prompt de comandos en el directorio donde reside ConsoleRead.exe
; Tipea lo siguiente en la línea de comandos:
; echo Hola! | ConsoleRead.exe
; Cuando se invoque una ventana de consola, el anterior comando muestra (hecho)el texto "Hola!"
; Pero en lugar de mostrar esto, el | avisa a la consola que el símbolo pipe indica el "STDIN stream del proceso ConsoleRead.exe"

If Not @Compiled Then

MsgBox(0, "", "Este script debe ser compilado en orden hará mostrar correctamente su funcionamiento.")

Exit -1

```
EndIf  
Local $data  
While True  
    $data &= ConsoleRead()  
    If @error Then ExitLoop  
    Sleep(25)  
WEnd  
MsgBox(0, "", "Recibido: " & @CRLF & @CRLF & $data)
```

ConsoleWrite

Escribe datos en la secuencia de STDOUT. Algunos editores de texto pueden leer dicha secuencia como otros programas.

ConsoleWrite ("dato")

Parámetros

dato	El dato para mostrar en la secuencia de salida. Este puede ser texto o binario.
------	---

Valor de Retorno

La cantidad de dato escrito. Si la escritura es binaria, el número de bytes escritos, si la escritura es textual, el número de caracteres escritos.

Comentarios

El propósito dicha función es escribir en la secuencia de STDOUT. Algunos editores de texto populares pueden leer dicha secuencia. Los script compilados como consola tiene una secuencia STDOUT.

Esta función no escribe en consola del DOS a menos que el script sea compilado como aplicación de consola.

Los caracteres son convertidos a ANSI antes de ser escritos.

Los datos binarios son escritos tan como aparecen. Estos no son convertidos a cadena. Para imprimir una representación hexadecimal de un datos binarios, use la función String() para un cambio explícito del dato a cadena.

Relativo

[ConsoleWriteError](#), [ConsoleRead](#)

Ejemplo

```
Local $var = "Test"  
ConsoleWrite("var=" & $var & @CRLF)  
; Correr esto en un editor de texto (eje. Scite) que pueda atrapar la consola de salida y verá  
como produce: "var=Test"
```

ConsoleWriteError

Escribe datos en la secuencia de STDERR. Algunos editores de texto pueden leer dicha secuencia como otros programas.

ConsoleWriteError ("dato")

Parámetros

dato	El dato para mostrar en la secuencia de salida. Este puede ser texto o binario.
------	---

Valor de Retorno

La cantidad de dato escrito. Si la escritura es binaria, el número de bytes escritos, si la escritura es textual, el número de caracteres escritos.

Comentarios

El propósito dicha función es escribir en la secuencia de STDERR. Algunos editores de texto populares pueden leer dicha secuencia. Los script compilados como consola tiene una secuencia STDERR.

Esta función no escribe en consola del DOS a menos que el script sea compilado como aplicación de consola.

Los caracteres son convertidos a ANSI antes de ser escritos.

Los datos binarios son escritos tan como aparecen. Estos no son convertidos a cadena. Para imprimir una representación hexadecimal de un datos binarios, use la función String() para un cambio explícito del dato a cadena.

Relativo

[ConsoleWrite](#), [ConsoleRead](#)

Ejemplo

```
Local $var = "Test"  
ConsoleWriteError("var=" & $var & @CRLF)  
; Correr esto en un editor de texto (eje. Scite) que pueda atrapar la consola de salida y verá  
como produce: "var=Test"
```

DirCopy

Copia un directorio conjuntamente con sus subdirectorios y archivos (Similar a xcopy).

DirCopy ("dir fuente", "dir destino" [, flag])

Parámetros

dir fuente	Ruta del directorio fuente (sin backslash"\\" al final). Eje. "C:\Path1"
dir destino	Ruta del directorio destino (sin backslash"\\" al final). Eje. "C:\Path_Copy"
flag	[opcional] determina la forma como son sobreescritos los ficheros si estos existen: 0 = (por defecto) no sobre escribir ficheros existentes 1 = sobre escribir ficheros existentes

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si existe un error durante la copia.

Comentarios

Si la estructura del directorio destino no existe, este será creado (si es posible).

Relativo

[DirRemove](#), [FileCopy](#)

Ejemplo

```
DirCopy(@MyDocumentsDir, "C:\Backups\MyDocs", 1)
```

DirGetSize

Devuelve el tamaño en bytes de un directorio dado.

DirGetSize ("ruta" [, flag])

Parámetros

ruta	La ruta del directorio para obtener el tamaño, Eje. "C:\Windows".
flag	[opcional] Determina el comportamiento del resultado de esta función, puede ser una combinación de lo siguiente: 0 = (por defecto) 1 = Modo extendido On -> devuelve un arreglo con información extendida (ver observaciones). 2 = No obtiene el tamaño de los archivos en los subdirectorios (modo recursivo Off)

Valor de Retorno

Con Éxitos Devuelve ≥ 0 (el tamaño)

Al Fallar Devuelve -1 y fija @error a 1 si la ruta no existe.

Comentarios

Si el script es pausado entonces esta función se detendrá; continuará una vez que el script halla salido de la pausa!

Si usted elige el modo extendido entonces se devolverá un arreglo lineal con la siguiente configuración:

\$arreglo[0] = Tamaño

\$arreglo[1] = Contador de ficheros

\$arreglo[2] = Contador de carpetas

Relativo

Ninguno.

Ejemplo

```
$Size = DirGetTamaño(@HomeDrive)
Msgbox(0,"","Tamaño(MegaBytes):" & Round($Size / 1024 / 1024))
```

```
$Size = DirGetTamaño(@WindowsDir, 2)
Msgbox(0,"","Tamaño(MegaBytes):" & Round($Size / 1024 / 1024))
```

```
$timer = TimerInit()
$Size = DirGetTamaño("\10.0.0.1\h$",1)
$diff = Round(TimerDiff($timer) / 1000) ; Tiempo en segundos
If IsArray($Size) Then
```

```
Msgbox(0,"DirGetSize-Info","Tamaño(Bytes):" & $Size[0] & @LF_
& "Ficheros:" & $Size[1] & @LF & "Directorios:" & $Size[2] & @LF_
& "TimeDiff(Sec):" & $diff)
```

EndIf

DirMove

Mueve un directorio y todos sus subdirectorios y ficheros.

DirMove (dir fuente, dir destino [, flag])

Parámetros

dir fuente	Ruta del directorio fuente (sin backslash al final). Eje. "C:\Path1"
dir destino	Ruta del directorio destino (sin backslash al final). Eje. "C:\Path_Copy"
flag	[opcional] Determina si se han de sobrescribir los archivos existentes: 0 = (por defecto) no sobrescribe los archivos existentes 1 = sobrescribe los archivos existentes

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si ocurre un error en la operación.

Comentarios

Si la fuente y el destino se encuentran en volúmenes diferentes o rutas UNC entonces es usado una operación copy/delete para optimizar la operación.

Si el destino ya existe y parámetro de sobre escritura es dado entonces el directorio fuente será movido **hacia dentro** del directorio destino.

Debido a que AutoIt carece de una función "DirRename", use DirMove para renombrar una carpeta!

Relativo

[DirRemove](#), [FileMove](#)

Ejemplo

`DirMove(@MyDocumentsDir, "C:\Backups\MyDocs")`

DirRemove

Borra un directorio/carpeta.

DirRemove ("ruta" [, recursivo])

Parámetros

ruta	Ruta del directorio a borrar.
recursivo	[opcional] use esta opción para especificar si desea borrar los subdirectorios también. 0 = (por defecto) no borra los archivos y subdirectorios 1 = Borra archivos y subdirectorios (semejante al comando del DOS: DelTree)

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si existe un error en la operación (o si el directorio no existe).

Comentarios

Algunos atributos de directorios pueden hacer imposible la operación de borrar.

Relativo

[DirCreate](#), [FileRecycle](#), [DirCopy](#), [DirMove](#), [FileDelete](#)

Ejemplo

; Borra C:\Test1 y todos los subdirectorios y ficheros
`DirRemove("C:\Test1", 1)`

DriveGetDrive

Devuelve un arreglo devolviendo las unidades enumeradas.

DriveGetDrive ("Tipo")

Parámetros

Tipo	Tipo de unidad a encontrar: "ALL", "CDROM", "REMOVABLE", "FIXED", "NETWORK", "RAMDISK", o
------	--

	"UNKNOWN"
--	-----------

Valor de Retorno

- Con Éxitos Devuelve un arreglo de cadenas (letra de la unidad seguida por una coma) de las unidades encontradas. El elemento de índice cero contiene el número de unidades encontradas.
- Al Fallar Devuelve "" y establece el valor de @error en 1.

Comentarios

Ninguno

Relativo

[DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveGetType](#), [DriveSetLabel](#), [DriveSpaceFree](#), [DriveSpaceTotal](#), [DriveStatus](#)

Ejemplo

```
$var = DriveGetDrive( "all" )
If NOT @error Then
    MsgBox(4096, "", "Encontrado " & $var[0] & " Unidades")
    For $i = 1 to $var[0]
        MsgBox(4096,"Unidad " & $i, $var[$i])
    Next
EndIf
```

DriveGetFileSystem

Devuelve el Tipo de Sistema de Archivo de una unidad.

DriveGetFileSystem ("ruta")

Parámetros

ruta	Ruta de la unidad para recibir la información.
------	--

Valor de Retorno

Con Éxitos Devuelve el Tipo de Sistema de Archivo como una cadena; ver tabla más abajo.

Al Fallar Establece @error a 1.

Valor de Retorno	Interpretación
1 (numérico)	Unidad NO contiene medio (CD, Floppy, Zip) o el medio esta sin formatear (RAW).
"FAT"	Sistema de archivos típicos para unidades menores de ~500 MB tales como Floppy, RAM disks, USB "pen", etc.
"FAT32"	Sistema de archivos típicos para unidades de disco duros en Windows 9x/Me.
"NTFS"	Sistema de archivos típicos para unidades de disco duros en Windows NT/2000/XP.
"NWFS"	Sistema de archivos típicos para servidores de archivos Novell Netware.
"CDFS"	Típicamente indica un CD (o una imagen ISO montado como unidad CD virtual).
"UDF"	Típicamente indica un DVD.

Comentarios

La lista completa de valores de retorno puede estar incompleta.

Relativo

[DriveGetDrive](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveGetType](#), [DriveSetLabel](#), [DriveSpaceFree](#), [DriveSpaceTotal](#), [DriveStatus](#)

Ejemplo

```
$var = DriveGetFileSystem( "c:\\" )
MsgBox(4096,"Tipo de Sistema de Archivo:", $var)
```

DriveGetLabel

Devuelve la Etiqueta de Volumen de una unidad, si lo posee.

DriveGetLabel ("ruta")

Parámetros

ruta	Ruta de la unidad para obtener la información.
------	--

Valor de Retorno

Con Éxitos Devuelve la etiqueta de Volumen de una unidad como una cadena.

Al Fallar Establece @error a 1.

Comentarios

Ninguno

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetSerial](#), [DriveGetType](#), [DriveSetLabel](#),
[DriveSpaceFree](#), [DriveSpaceTotal](#), [DriveStatus](#)

Ejemplo

```
$var = DriveGetLabel( "c:\\" )
MsgBox(4096,"Etiqueta de volumen: ",$var)
```

DriveGetSerial

Devuelve el Número de Serie de una unidad.

DriveGetSerial ("ruta")

Parámetros

ruta	Ruta de la unidad para obtener la información.
------	--

Valor de Retorno

Con Éxitos Devuelve el Número de Serie de una unidad como una cadena.

Al Fallar Establece @error a 1.

Comentarios

El valor devuelto no es el número de serial de hardware como el encontrado en la etiqueta de la unidad, este es el ID del Volumen de Windows para la unidad.

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetType](#), [DriveSetLabel](#),
[DriveSpaceFree](#), [DriveSpaceTotal](#), [DriveStatus](#)

Ejemplo

```
$var = DriveGetSerial( "c:\" )
MsgBox(4096, "Número de serial: ", $var)
```

DriveGetType

Devuelve tipo de unidad.

DriveGetType ("ruta")

Parámetros

ruta	Ruta de la unidad para obtener información.
------	---

Valor de Retorno

Con Éxitos Devuelve el tipo de unidad: "Unknown", "Removable", "Fixed", "Network",
"CDROM", "RAMDisk"

Al Fallar Devuelve "" y establece el valor de @error en 1.

Comentarios

La lista completa de valores puede estar incompleta.

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveSetLabel](#),
[DriveSpaceFree](#), [DriveSpaceTotal](#), [DriveStatus](#), [CDTray](#)

Ejemplo

```
$var = DriveGetType( "c:\" )
MsgBox(4096, "Tipo de Unidad:", $var)
```

DriveMapAdd

Mapea una unidad de red.

DriveMapAdd ("dispositivo", "recurso remoto" [, flags [, "usuario" [, "contraseña"]]])

Parámetros

dispositivo	El dispositivo a mapear, Por ejemplo "O:" o "LPT1:". Si usted pasa una cadena vacía por este parámetro se realiza una conexión pero ningún dispositivo es mapeado. Si usted especifica "*" una letra de unidad no en uso será automáticamente seleccionada.
recurso remoto	El recurso remoto para conectar en la forma "\\\server\share".
flags	[opcional] Una combinación de los siguientes valores: 0 = por defecto 1 = Mapeo persistente 8 = Mostrar el diálogo de autenticación.
usuario	[opcional] El nombre de usuario para autenticar la conexión. En la forma "NombreUsuario" o "dominio\NombreUsuario".
contraseña	[opcional] La contraseña para ser usada en la conexión.

Valor de Retorno

Con Éxitos Devuelve 1. (Ver comentarios)

Al Fallar	Devuelve 0 si el nuevo mapeo no puede ser creado y fija @error (ver más abajo). (Ver Comentarios)
-----------	---

Comentarios

Cuando la función falla (devuelve 0) @error contiene la siguiente información extendida:

1 = Indefinido / otro error. @extended es establecido con el retorno de la API Windows

2 = Acceso al recurso remoto denegado

3 = El dispositivo ya ha sido asignado

4 = Nombre de dispositivo inválido

5 = Recurso remoto no válido

6 = Contraseña no válida

Nota: Cuando usamos "*" para el parámetro *dispositivo* la letra de la unidad seleccionada será devuelto en lugar de 1 o 0, Eje. "U:". Si ocurre un error utilizando "*" entonces una cadena vacía "" será devuelta.

Si definimos el usuario/contraseña será presentado al computador remoto para validar la credencial.

Relativo

[DriveMapDel](#), [DriveMapGet](#)

Ejemplo

; Mapea la unidad X a \\myserver\stuff usando el usuario actual
`DriveMapAdd("X:", "\\myserver\stuff")`

; Mapea la unidad X a \\myserver2\stuff2 usando el usuario "jon" desde "domainx" con clave "tickle"
`DriveMapAdd("X:", "\\myserver2\stuff2", 0, "domainx\jon", "tickle")`

DriveMapDel

Desconecta una unidad de red.

DriveMapDel ("unidad")

Parámetros

unidad	El dispositivo a desconectar, Eje. "O:" o "LPT1:".
--------	--

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la operación no es satisfactoria.

Comentarios

Si la letra de una unidad conectada no existe puede utilizar el nombre de la conexión para desconectarla, Eje. \\server\share

Relativo

[DriveMapAdd](#), [DriveMapGet](#)

Ejemplo

; Mapea la unidad X a \\myserver\stuff usando el usuario actual
DriveMapAdd("X:", "\\myserver\stuff")

; Desconectando
DriveMapDel("X:")

DriveMapGet

Recupera información detallada de una unidad mapeada.

DriveMapGet ("dispositivo")

Parámetros

dispositivo	La letra del dispositivo (unidad o impresora) para consultar, Eje. "O:" o "LPT1:"
-------------	---

Valor de Retorno

Con Éxitos	Devuelve detalles del mapeo, Eje. \\server\share
Al Fallar	Devuelve una cadena vacía "" y establece el valor de @error en 1.

Comentarios

Ninguno

Relativo

[DriveMapAdd](#), [DriveMapDel](#)

Ejemplo

```
; Mapea la unidad X a \\myserver\stuff usando el usuario actual
DriveMapAdd("X:", "\\myserver\stuff")
```

```
; Obtiene detalles del mapeado
MsgBox(0, "Drive X: es mapeado a", DriveMapGet("X:"))
```

DriveSetLabel

Establece la Etiqueta de Volumen de una unidad.

DriveSetLabel ("ruta", "etiqueta")

Parámetros

ruta	Ruta de la unidad a cambiar.
etiqueta	Nueva etiqueta de volumen de la unidad. (11 caracteres es el máximo usado)

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

Muchas unidades de disco duro tienen una etiqueta de volumen de 11 caracteres como máximo, y DriveSetLabel fallará si la longitud es excedida. También, las etiquetas en particiones FAT32 tiende a revertir todas las letras a capitales.

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveGetType](#),
[DriveSpaceFree](#), [DriveSpaceTotal](#), [DriveStatus](#)

Ejemplo

```
DriveSetLabel("C:\", "Nueva_etiqueta")
```

DriveSpaceFree

Devuelve el espacio libre en disco(en Megabytes)de una ruta en Megabytes.

DriveSpaceFree ("ruta")

Parámetros

ruta	Ruta de la unidad para recibir la información.
------	--

Valor de Retorno

Con Éxitos Devuelve el espacio libre en disco en Megabytes como número flotante (decimal).

Al Fallar Devuelve 0 y establece el valor de @error en 1.

Comentarios

DriveSpaceFree puede trabajar cuando es dada una ruta completa (que existe). Sin embargo, una ruta de archivo no es válida.

Use la función Round() para redondear el resultado de acuerdo a la precisión deseada.

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveGetType](#),
[DriveSetLabel](#), [DriveSpaceTotal](#), [DriveStatus](#)

Ejemplo

```
$var = DriveSpaceFree( "c:\\" )  
MsgBox(4096, "Espacio libre en C:", $var & " MB")
```

DriveSpaceTotal

Devuelve el espacio total libre en disco de una ruta en Megabytes.

DriveSpaceTotal ("ruta")

Parámetros

ruta	Ruta de la unidad para recibir la información.
------	--

Valor de Retorno

Con Éxitos Devuelve el espacio libre en disco en Megabytes como número flotante (decimal).

Al Fallar Establece @error a 1.

Comentarios

DriveSpaceFree puede trabajar cuando es dada una ruta completa (que existe). Sin embargo, una ruta de archivo no es válida.

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveGetType](#),
[DriveSetLabel](#), [DriveSpaceFree](#), [DriveStatus](#), [FileGetSize](#)

Ejemplo

```
$var = DriveSpaceTotal( "c:\\" )  
MsgBox(4096, "Espacio total en C:", $var & " MB")
```

DriveStatus

Devuelve el estado de una unidad como una cadena.

DriveStatus ("ruta")

Parámetros

ruta	Ruta de la unidad para recibir la información.
------	--

Valor de Retorno

Valor	Interpretación
UNKNOWN	Unidad puede ser formateada (RAW).
READY	Típico de discos duros y unidades que contienen medios removibles.
NOTREADY	Típico de discos floppy y unidades de CD que no son medios.
INVALID	Puede indicar que la letra de unidad no existe o unidad de red mapeada no es accesible.

Comentarios

La lista completa de valores de retorno puede estar incompleta.

DriveStatus puede trabajar cuando es dada una ruta completa (que existe). Sin embargo, una ruta de archivo no es válida.

Relativo

[DriveGetDrive](#), [DriveGetFileSystem](#), [DriveGetLabel](#), [DriveGetSerial](#), [DriveGetType](#),
[DriveSetLabel](#), [DriveSpaceFree](#), [DriveSpaceTotal](#), [CDTray](#), [FileExists](#)

Ejemplo

```
$var = DriveStatus( "c:\\" )
MsgBox(4096,"Estatus",$var)
```

FileChangeDir

Cambia el directorio actual de trabajo.

FileChangeDir ("ruta")

Parámetros

ruta	La ruta para hacer el actual directorio de trabajo.
------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el directorio de trabajo no cambia.

Comentarios

Ninguno

Relativo

[@WorkingDir](#)

Ejemplo

FileChangeDir(@WindowsDir)

FileClose

Cierra un fichero de texto previamente abierto.

FileClose (Apuntador_Fichero)

Parámetros

Apuntador_Fichero	El apuntador de un fichero, devuelto por una llamada previa a FileOpen.
-------------------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el Apuntador_Fichero no es válido.

Comentarios

Al finalizar, AutoIt automáticamente cierra cualquier archivo abierto, pero utilizar FileClose es una buena idea.

Esta función es también usada cerrar búsquedas de apuntadores (handles) devueltos por FileFindFirstFile().

Relativo

[FileFindFirstFile](#), [FileOpen](#), [FileFindNextFile](#), [FileFlush](#)

Ejemplo

```
$handle = FileOpen("test.txt", 0)
```

```
FileClose($handle)
```

FileCopy

Copia uno o más ficheros.

FileCopy ("fuente", "destino" [, flag])

Parámetros

fuente	La ruta fuente de los ficheros a copiar. Comodines son soportados.
destino	La ruta de destino de los ficheros a copiar.
flag	[opcional] Determina si son sobrescritos los ficheros existentes. Puede ser una combinación de lo siguiente: 0 = (por defecto) no sobrescribe los ficheros existentes

	<p>1 = sobrescribe los ficheros existentes 8 = Crea una estructura del directorio destino de no existir este (Ver observaciones).</p>
--	--

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

El directorio destino debe existir, excepto cuando la opción es puesta a '8'.

Por instancia la opción en combinación '9' (1 + 8) sobrescribe los ficheros existentes y prechequea si el directorio destino existe de lo contrario lo crea automáticamente.

Ver [FileFindFirstFile](#) para una discusión sobre comodines.

Algunos atributos de archivos pueden hacer la sobre escritura imposible.

Relativo

[FileMove](#), [FileDelete](#), [DirCopy](#), [DirCreate](#)

Ejemplo

`FileCopy("C:*.au3", "D:\mydir*.*")`

; Método para copiar una carpeta (con su contenido)

`DirCreate("C:\new")`

`FileCopy("C:\old*.*", "C:\new\")`

`FileCopy("C:\Temp*.txt", "C:\Temp\TxtFiles\", 8)`

; CORRECTO - 'TxtFiles' Es ahora el directorio blanco y los nombres del archivo son dados por los nombres de la fuente

`FileCopy("C:\Temp*.txt", "C:\Temp\TxtFiles\", 9); Flag = 1 + 8 (overwrite + create target directory structure)`

; Copia los ficheros txt desde la fuente al directorio blanco y sobrescribe archivos de blanco con mismo nombre

FileCreateNTFSLink

Crea un hardlink NTFS para un archivo o directorio.

FileCreateNTFSLink ("fuente", "hardlink" [, flag])

Parámetros

fuente	La ruta del recurso para crear el hardlink.
hardlink	Ruta del hardlink.
flag	[opcional] Determina si son sobrescritos los ficheros existentes. Puede ser una combinación de lo siguiente: 0 = (por defecto) no sobrescribe los ficheros existentes 1 = no sobrescribe los ficheros existentes

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

El directorio destino debe existir.

Esta función funciona solamente en volúmenes con sistema de archivos NTFS.

Si la fuente es un fichero, el hardlink debe estar en el mismo volumen.

Si la fuente es un directorio un volumen puede ser permitido.

FileDelete o FileMove puede ser usado en hardlink.

Para manejar el enlace con el explorador usted puede usar la extensión shell [NTFSLink](#)

Relativo

[FileCreateShortcut](#)

Ejemplo

```
FileChangeDir(@ScriptDir)
DirCreate('directorio')
FileWriteLine("test.txt", "test")
MsgBox(0, "Hardlink", FileCreateNTFSLink("test.txt", "dir\test.log", 1))
```

FileCreateShortcut

Crea un acceso directo (.lnk) a un fichero.

FileCreateShortcut ("fichero", "lnk" [, "workdir" [, "argumentos" [, "descripción" [, "icono" [, "hotkey" [, icono número [, estado]]]]]]])

Parámetros

fichero	La ruta completa del nombre del archivo del cual se va a crear al acceso directo.
lnk	La ruta completa del nombre del archivo para crear el enlace.
workdir	[opcional] Directorio de trabajo.
argumentos	[opcional] Argumentos de archivo adicionales.
descripción	[opcional] Descripción del fichero.
icono	[opcional] Ruta completa del ícono o del fichero de íconos a utilizar.
hotkey	[opcional] Combinación de teclas - Con el mismo formato de Send().
ícono número	[opcional] La instancia del ícono a utilizar (usualmente 0)
estado	[opcional] El estado del acceso directo al ser este ejecutado. Puede ser @SW_SHOWNORMAL, @SW_SHOWMINNOACTIVE o @SW_SHOWMAXIMIZED

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si lnk no puede ser creado.

Comentarios

Los Hotkeys para accesos en Windows son de la siguiente forma: Ctrl+Alt+X, Ctrl+Shift+X, Shift+Alt+X, Ctrl+NumPadKey, o Alt+NumPadKey donde X representa una letra, número, puntuación, o tecla de función. Si usted especifica una forma no válida, Windows usa por defecto Ctrl+Alt

Note que Windows distingue número del teclado numérico de números regulares y teclas de puntuación. FileCreateShortcut le permite a usted crear accesos Ctrl+X y Alt+X (que Windows permite solamente cuando X es un NumPadKey); Sin embargo, usted debe evitar estos asignamientos para evitar conflictos con combinaciones de teclas(Hotkey). Windows prohíbe ESC, ENTER, TAB, SPACEBAR, PRINT SCREEN, SHIFT, o BACKSPACE para usar combinaciones de teclas(Hotkey).

FileCreateShortcut no requiere un blanco puntual, directorio de trabajo, icono, combinación de teclas (Hotkey) en orden para crear "satisfactoriamente" fichero LNK; Sin embargo, la ruta destino del archivo LNK debe ser válido! Si la combinación de teclas ya esta en uso, su nuevo acceso directo toma precedencia. También, si usted crea un acceso directo con alguna ruta\nombre de un acceso ya existente, este sobrescribe el antiguo y toma precedencia.

Relativo

[FileGetShortcut](#), [FileCreateNTFSLink](#)

Ejemplo

```
; establece el método de teclado ctrl+alt+t
FileCreateShortcut(@WindowsDir & "\Explorer.exe", @DesktopDir & "\Shortcut
Test.lnk", @WindowsDir, "/e,c:\", "This is an Explorer link;-)", @SystemDir & "\shell32.dll", "^It",
"15", @SW_MINIMIZE)
```

FileDelete

Elimina uno o más ficheros.

FileDelete ("ruta")

Parámetros

ruta	La ruta de los ficheros a borrar. Comodines son soportados.
------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si los archivos no son borrados o no existen.

Comentarios

Note: Si la "ruta" pasada a FileDelete es una carpeta, los archivos dentro de la carpeta serán borrados como si usted especificara el comodín *.*.

Ver [FileFindFirstFile](#) para una discusión sobre los comodines.

Algunos atributos de archivo pueden hacer imposible la eliminación.

Relativo

[FileCopy](#), [FileMove](#), [FileRecycle](#), [DirRemove](#), [FileRecycleEmpty](#)

Ejemplo

`FileDelete("D:*.tmp")`

FileExists

Chequea si un archivo o directorio existe.

`FileExists ("ruta")`

Parámetros

Ruta	El directorio o archivo a revisar.
------	------------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ruta/fichero no existe.

Comentarios

FileExists devuelve 0 si usted especifica una unidad floppy que no contiene ningún disco.

Relativo

[FileGetAttrib](#), [DriveStatus](#)

Ejemplo

```
If FileExists("C:\autoexec.bat") Then
    MsgBox(4096, "fichero C:\autoexec.bat", "Existe")
Else
    MsgBox(4096, "fichero C:\autoexec.bat", "no existe")
EndIf

If FileExists("C:\") Then
    MsgBox(4096, "Directorio C:\ ", "Existe")
Else
    MsgBox(4096, "Directorio C:\ ", "No existe")
EndIf

If FileExists("D:") Then
    MsgBox(4096, "Unidad D: ", "Existe")
Else
    MsgBox(4096, "Unidad D: ", "No existe")
EndIf
```

FileFindFirstFile

Devuelve un identificador de búsqueda de acuerdo con una cadena de búsqueda.

FileFindFirstFile ("NombreFichero")

Parámetros

NombreFichero	Cadena de búsqueda del fichero. (* y ? comodines son soportados)
---------------	--

Valor de Retorno

Con Éxitos Devuelve un identificador "handle" de búsqueda para usar subsecuentemente con funciones FileFindNextFile.

Al Fallar Devuelve -1 si ocurre un error. Si la carpeta esta vacía @error es fijado a un 1.

Comentarios

La cadena de búsqueda no es sensible a mayúscula y minúsculas. Comodines: En general, * denota cero o más caracteres, y ? denota cero o un carácter. Si su cadena de búsqueda del archivo contiene solamente comodines (o es "*.*"), entonces ver el ejemplo más abajo para el valor de retorno!

Usted puede utilizar solamente comodines con una parte del nombre del archivo o la extensión ej. a*.b?.

?? es equivalente a * (no descrito en la documentación de Microsoft). Cuando se utilizan extensiones de 3 caracteres cualquier extensión que comience con 3 caracteres será coincidente, ej. "* .log" coincide con "test.log_1". (no descrito en la documentación de Microsoft).

Cuando usted tiene una búsqueda finalizada con funciones FileFind... Usted puede invocar FileClose() para liberar el identificador de búsqueda.

Los nombres de los directorios son retornados de acuerdo a los comodines si los hay.

Relativo

[FileClose](#), [FileFindNextFile](#)

Ejemplo

```
; Muestra los nombres de ficheros y todos los ficheros en el directorio actual
$search = FileFindFirstFile("*. *")

; Chequea si la búsqueda a sido satisfactoria
If $search = -1 Then
    MsgBox(0, "Error", "No ficheros/directorios coincidentes con el criterio anterior")
    Exit
EndIf

While 1
    $file = FileFindNextFile($search)
    If @error Then ExitLoop

    MsgBox(4096, "Fichero:", $file)
WEnd

; Cierra el identificador de la búsqueda
FileClose($search)
```

FileFlush

Limpia el buffer de una fichero en disco.

FileFlush (identificador)

Parámetros

identificador	Un identificador a un fichero previamente abierto con FileOpen().
---------------	---

Valor de Retorno

Con Éxitos: Devuelve true si el buffer fué limpiado (o no necesita ser limpiado).

Al Fallar: Devuelve false.

Comentarios

Un fichero es limpiado cuando el identificador es cerrado o cuando el buffer interno de Windows esta lleno. Esta función fuerza a un limpiado inmediato del buffer.

Esta función puede ser usada solamente con identificadores devueltos por FileOpen().

Relativo

[FileClose](#), [FileOpen](#), [FileWrite](#), [FileWriteLine](#), [FileSetPos](#)

Ejemplo

```
Local Const $sFile = "test.txt"
Local $hFile = FileOpen($sFile, 1)

; Chequea si el fichero es correctamente abierto para escritura:
If $hFile = -1 Then
    MsgBox(0, "Error", "Unable to open file.")
    Exit
EndIf

;Escribiendo algo en el fichero.
FileWriteLine($hFile, "Line1")

; ejecutar notepad para mostrar que el fichero esta vacío debido a que no se ha flusheado
; todavía.
RunWait("notepad.exe " & $sFile)

; Flusheo del fichero en el disco.
FileFlush($hFile)

; ejecutar notepad para mostrar que el contenido del fichero es ahora flusheado en el disco.
RunWait("notepad.exe " & $sFile)
```

```
; Cierra el identificador.  
FileClose($hFile)  
  
; Limpia el fichero temporal  
FileDelete($sFile)
```

FileGetAttrib

Devuelve una cadena codificada conteniendo los atributos de un fichero.

FileGetAttrib (nombreFichero)

Parámetros

nombreFichero	nombreFichero (o directorio) a revisar.
---------------	---

Valor de Retorno

Con Éxitos Devuelve una cadena codificada conteniendo los atributos de un archivo.

Al Fallar Devuelve "" (cadena vacía) y establece el valor de @error en 1.

Comentarios

La cadena devuelta puede ser una combinación de estas letras "RASHNDOCT":

"R" = READONLY
"A" = ARCHIVE
"S" = SYSTEM
"H" = HIDDEN
"N" = NORMAL
"D" = DIRECTORY
"O" = OFFLINE
"C" = COMPRESSED (compresión NTFS , no compresión ZIP)
"T" = TEMPORARY

Relativo

[FileGetTime](#), [FileSetAttrib](#), [FileExists](#), [FileGetSize](#), [FileSetTime](#)

Ejemplo

```
$attrib = FileGetAttrib("c:\boot.ini")
```

```

If @error Then
    MsgBox(4096, "Error", "No se puede obtener los atributos.")
    Exit
Else
    If StringInStr($attrib, "R") Then
        MsgBox(4096, "", "Fichero de solo lectura.")
        EndIf
    EndIf

; Muestra toda la información de los atributos como texto en la forma de un
; Arreglo confiando en el echo de que cada letra mayúscua es única
; Entienda como este trabajo es un buen ejercicio de cadena...
$input = StringSplit("R,A,S,H,N,D,O,C,T", ",")
$output = StringSplit("Read-only /, Archive /, System /, Hidden /" & _
    ", Normal /, Directory /, Offline /, Compressed /, Temporary /", ",")
For $i = 1 to 9
    $attrib = StringReplace($attrib, $input[$i], $output[$i], 0, 1)
    ; último parámetro en StringReplace significa caso sensitivo
Next
$attrib = StringTrimRight($attrib, 2) ;removiendo los slash
MsgBox(0, "Atributos totales del fichero:", $attrib)

```

FileGetEncoding

Determina la codificación de texto utilizada en un fichero.

FileGetEncoding ("filehandle/filename" [, modo])

Parámetros

filehandle/filename	El identificador del fichero, returnedo de una llamada previa de FileOpen. Alternativamente puede utilizar una cadena con la ruta del fichero como primer parámetro.
modo	[opcional] El modo de detección UTF8 para usar. 1 = Chequea todo el fichero para secuencias UTF8 (por defecto) 2 = Chequea la primera parte del fichero para secuencias UTF8 (igual como FileOpen usa por defecto)

Valor de Retorno

Con Éxitos: Devuelve el fichero codificado usando valores similares para la función
FileOpen:

0 = ANSI

32 = UTF16 LittleEndian.

64 = UTF16 BigEndian.

128 = UTF8 (with BOM).

256 = (sin BOM).

Al Fallar: Devuelve -1.

Comentarios

Si un nombre de fichero es dado es vez de un identificador de fichero - el fichero será abierto y cerrado durante la llamada a la función.

Nota: No mezcle nombres de fichero e identificadores de ficheros, ej., no abra un fichero con FileOpen y luego use un nombre de fichero en esta función. Use cualquiera de ambos métodos de referenciar el fichero, no ambos!

Si un identificador de fichero es usado entonces el parámetro "modo" no tendrá efecto - el modo usado por FileOpen tomará precedencia.

Relativo

[FileOpen](#), [FileRead](#), [FileReadLine](#), [FileWrite](#), [FileWriteLine](#)

FileGetLongName

Devuelve el nombre completo (dirección+nombre) de una ruta dada.

FileGetLongName (fichero [, flag])

Parámetros

fichero	full ruta y archivo nombre a convertir.
flag	[opcional] si es 1 el archivo puede tener directorio relativo, Eje. "..\fichero.txt"

Valor de Retorno

- Con Éxitos Devuelve el nombre completo de un archivo.
- Al Fallar Devuelve el parámetro y establece el valor de @error en 1.

Comentarios

Ninguno

Relativo

[FileGetShortName](#)

Ejemplo

```
$a = FileGetLongName(@HomeDrive & "\PROGRA~1\")
msgbox(0,"nombre largo de fichero", $a)
;$a es probablemente "x:\Program Files"
```

FileGetPos

Devuelve la posición actual en un fichero

FileGetPos (identificador)

Parámetros

identificador	Un identificador a un fichero previamente abierto con FileOpen().
---------------	---

Valor de Retorno

Con Éxitos: Devuelve la posición del desplazamiento (offset) a partir del inicio del fichero (Primer índice es 0).

Al Fallar: Devuelve 0 y establece @error.

Comentarios

El retorno al fallar es 0 pero 0 es también una posición válida de fciehro así que consulte @error para determinar la condición de error.

Relativo

[FileSetPos](#), [FileRead](#), [FileReadLine](#), [FileWrite](#), [FileWriteLine](#), [FileOpen](#)

Ejemplo

```
#include <Constants.au3>

Local Const $sFile = "test.txt"
Local $hFile = FileOpen($sFile, 2)

; Chequea si el fichero es correctamente abierto para escritura.
If $hFile = -1 Then
    MsgBox(0, "Error", "Unable to open file.")
    Exit
EndIf

;Escribiendo algo en el fichero.
FileWriteLine($hFile, "Line1")
FileWriteLine($hFile, "Line2")
FileWriteLine($hFile, "Line3")

; Flusheo del fichero en el disco.
FileFlush($hFile)

; Chequeando la posición en el fichero y probando leer el contenido de la posición actual.
MsgBox(0, "", "Position: " & FileGetPos($hFile) & @CRLF & "Data: " & @CRLF &
FileRead($hFile))

; Ahora, ajusta la posición al inicio del fichero.
Local $n = FileSetPos($hFile, 0, $FILE_BEGIN)

; Chequeando la posición del fichero y probando leer el contenido de la posición actual.
MsgBox(0, "", "Position: " & FileGetPos($hFile) & @CRLF & "Data: " & @CRLF &
FileRead($hFile))

; Cierra el identificador.
FileClose($hFile)

; Limpia el fichero temporal
FileDelete($sFile)
```

FileGetShortcut

Devuelve detalles de un acceso directo.

FileGetShortcut ("lnk")

Parámetros

Ink	Ruta completa del fichero de acceso directo.
-----	--

Valor de Retorno

Con Éxitos Devuelve un arreglo que contiene la información del acceso directo. Ver observaciones.

Al Fallar Establece @error en 1 si el acceso directo no puede ser accedido.

Comentarios

El arreglo devuelto por esta función es un arreglo unidimensional conteniendo los siguientes elementos:

\$arreglo[0] = Ruta del acceso directo
\$arreglo[1] = Directorio de trabajo
\$arreglo[2] = Argumentos
\$arreglo[3] = Descripción
\$arreglo[4] = Nombre de fichero del ícono
\$arreglo[5] = Índice del ícono
\$arreglo[6] = El estado del acceso directo (@SW_SHOWNORMAL, @SW_SHOWMINNOACTIVE, @SW_SHOWMAXIMIZED)

Relativo

[FileCreateShortcut](#)

Ejemplo

```
; Establece el método de teclado ctrl+alt+  
FileCreateShortcut(@WindowsDir & "\Explorer.exe", @DesktopDir & "\Shortcut  
Test.lnk", @WindowsDir, "/e,c:\", "This is an Explorer link;-)", @SystemDir & "\shell32.dll", "^!t",  
"15", @SW_MINIMIZE)  
  
; Lee en la dirección del método de teclado  
$details = FileGetShortcut(@DesktopDir & "\Shortcut Test.lnk")  
MsgBox(0, "Path:", $details[0])
```

FileGetShortName

Devuelve el nombre corto 8.3 de una dirección+nombre de archivo dado.

FileGetShortName (fichero [, flag])

Parámetros

fichero	Ruta completa del nombre del fichero de fichero a convertir.
flag	[opcional] si es 1 el archivo puede tener directorio relativo, Eje. "..\fichero.txt"

Valor de Retorno

Con Éxitos Devuelve el nombre corto 8.3 de una dirección+nombre de archivo.

Al Fallar Devuelve el parámetro y establece el valor de @error en 1.

Comentarios

El archivo necesita existir pero no hay posibilidad de detectar ~ si existen varios ficheros que tienen iguales los 8 primeros caracteres.

Relativo

[FileGetLongName](#)

Ejemplo

```
$a = FileGetShortName(@HomeDrive & "\Program Files")
msgbox(0,"long file name", $a)
;$a es probablemente "x:\PROGRA~1"
```

FileSize

Devuelve el tamaño de un archivo en bytes.

FileGetSize (nombreFichero)

Parámetros

nombreFichero	nombreFichero para revisar.
---------------	-----------------------------

Valor de Retorno

Con Éxitos Devuelve el tamaño de el archivo en bytes.

Al Fallar Devuelve 0 y fija el @error a 1.

Comentarios

No trabaja en directorios.

Dividir el resultado entre 1024 para obtener el equivalente en kilobyte, o divide entre 1048576 para obtener el equivalente en megabyte.

Relativo

[FileGetAttrib](#), [FileGetTime](#), [DriveSpaceTotal](#), [FileGetVersion](#)

Ejemplo

```
$size = FileGetSize("AutoIt.exe")
```

FileGetTime

Devuelve información del tiempo y la fecha para un fichero.

FileGetTime (nombreFichero [, opción [, formato]])

Parámetros

nombreFichero	Nombre del fichero para revisar.
opciones	[opcional] indica el timestamp (marca de tiempo) 0 = Modificado (por defecto) 1 = Creado 2 = Accedido
formato	[opcional] para especificar el tipo de retorno 0 = retorna un arreglo (por defecto)

	1 = retorna una cadena YYYYMMDDHHMMSS
--	---------------------------------------

Valor de Retorno

- Con Éxitos Devuelve un arreglo o cadena que contiene información del tiempo del archivo.
Ver observaciones.
- Al Fallar Devuelve 0 y establece el valor de @error en 1.

Comentarios

El arreglo es unidimensional conteniendo seis elementos:

\$arreglo[0] = año (cuatro dígitos)
\$arreglo[1] = mes (rango 01 - 12)
\$arreglo[2] = día (rango 01 - 31)
\$arreglo[3] = hora (rango 00 - 23)
\$arreglo[4] = min (rango 00 - 59)
\$arreglo[5] = seg (rango 00 - 59)

Note que los valores devueltos son normalizados con ceros.

Relativo

[FileGetSize](#), [FileGetAttrib](#), [FileGetVersion](#), [FileSetTime](#), [FileSetAttrib](#)

Ejemplo

```
$t = FileGetTime(@Windowsdir & "\Notepad.exe", 1)

If Not @error Then
    $yyyymd = $t[0] & "/" & $t[1] & "/" & $t[2]
    MsgBox(0, "Creación de la fecha en notepad.exe", $yyyymd)
EndIf
```

FileGetVersion

Devuelve información de la versión del "fichero".

FileGetVersion (nombreFichero [, "stringname"])

Parámetros

nombreFichero	Nombre del fichero para revisar.
stringname	[opcional] nombre del campo para intentar recobrar la información desde el encabezado de información de versión del archivo.

Valor de Retorno

Con Éxitos Devuelve una cadena conteniendo la información de la versión , Eje. "3.0.81.0".

Al Fallar Devuelve "0.0.0.0" si no hay información de versión(o otro error) o "" cuando recupera un stringname, y establece el valor de @error en 1.

Comentarios

stringname puede ser básicamente uno de estos :

Comments, InternalName, ProductName, CompanyName, LegalCopyright, ProductVersion, FileDescription, LegalTrademarks, PrivateBuild,FileVersion, OriginalFilename, SpecialBuild

O uno especial "CompiledScript" el cual es establecido para un script compilado. FileGetVersion(@AutoltExe, "CompiledScript") devolverá "Autolt v3 Script : 3, 2, 1, 2".

Otro stringname especial es "DefaultLangCodepage" puede ser usado para recuperar el language y codepage por defecto.

El language y codepage puede ser usado si lo necesita para diferenciar el "stringname" ej. "080904b0\Comments" (ver MSDN StringFileInfo en la función VerQueryValue).

Relativo

[FileGetSize](#), [FileGetTime](#)

Ejemplo

```
$ver = FileGetVersion("Explorer.exe")
MsgBox(0, "Versión del Explorer", $ver)
```

FileInstall

Incluye e instala un archivo con el archivo compilado.

FileInstall ("fuente", "destino" [, flag])

Parámetros

fuente	La ruta fuente del archivo para compilar. Esto debe ser una cadena literal; no puede ser una variable o el resultado de una llamada de función. esto puede ser una dirección relativa (usando .\ o ..\ en la ruta) para el fichero fuente (.au3).
destino	El directorio destino con backslash si se usa hasta el directorio. Este puede ser una variable.
flag	[opcional] Determina si son sobrescritos los archivos existentes. 0 = (por defecto) no sobrescribe los ficheros existentes 1 = sobrescribe los ficheros existentes

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

La función FileInstall es designada para incluir archivos en un script de AutoIt compilado. Estos archivos incluidos pueden entonces ser "extraídos" durante la ejecución del script compilado. Tenga presente que los archivos como imágenes pueden incrementar el tamaño de un script compilado.

El archivo fuente debe ser una cadena de texto y **no** puede ser una variable, un cálculo o una llamada de función. El fichero debe poderse encontrar durante la compilación, sin embargo, variables, cálculos y llamadas de funciones no lo pueden resolver mientras el script este corriendo sobre sí mismo, después de la compilación, haciendo idanecuado definir el fichero fuente.

Cuando esta función es usada de un script no-compilado, una operación de copiar es realizada. (para probar fácilmente en fase de pre-compilación).

Los archivos pueden mantener su fecha original de creación cuando se instalan.

La ruta del directorio de destino debe realmente existir antes de que esta función sea llamada, o FileInstall fallará, devolviendo 0 y no creando el fichero, o la dirección. Ver DirCreate() para información acerca de la creación de directorios.

Los atributos de fichero de un fichero existente pueden prevenir que este sea sobrescrito. Use FileDelete() o FileSetAttrib() para asegurarse que el fichero pueda ser instalado sin problemas.

Relativo

[DirCreate](#), [FileDelete](#), [FileSetAttrib](#)

Ejemplo

```
; Incluye un bitmap encontrado en "C:\test.bmp" con el programa compilado y colocarlo en  
"D:\mydir\test.bmp" cuando este es ejecutado  
$b = True  
If $b = True Then FileInstall("C:\test.bmp", "D:\mydir\test.bmp")
```

FileMove

Mueve uno o más directorios.

FileMove ("fuente", "destino" [, flag])

Parámetros

fuente	La ruta fuente y el nombre del fichero a mover. (comodines * son soportados)
destino	La ruta destino y el nombre del fichero movidos. (comodines * son soportados)
flag	[opcional] Determina si los archivos existentes serán sobrescritos. Puede ser una combinación de lo siguiente: 0 = (por defecto) No sobrescribe los ficheros existentes 1 = comodines * son soportados 8 = Crea una estructura de directorio destino si este no existe (Ver comentarios).

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si *fuente* no puede ser movido o si *destino* ya existe y flag=0.

Comentarios

Si las rutas fuente y destino son en diferentes volúmenes una operación de copiar y borrar es ejecutada para mejorar la operación de mover.

Debido a que Autolt carece de una función "FileRename", use FileMove para renombrar un fichero!

El directorio destino debe existir, excepto cuando se usa la opción '8'.

Por instancia la combinación '9' (1 + 8) sobrescribe los ficheros existentes y pre-chequea si el directorio destino no existe y lo crea automáticamente.

Algunos atributos de archivo pueden hacer imposible la operación de mover.

Relativo

[FileCopy](#), [FileDelete](#), [FileRecycle](#), [DirMove](#)

Ejemplo

FileMove("C:\foo.au3", "D:\mydir\bak.au3")

*; Segundo ejemplo:
; usando la bandera '1' (sobreescibiendo) y '8' (Autocreando la estructura del
directorioblanco) juntamente
; mueve todos los ficheros txt desde temp a txtfiles y prechequea si
; la estructura del directorio blanco existe, si no, es creado automáticamente.
FileMove(@TempDir & "*.txt", @TempDir & "\TxtFiles\", 9)*

FileOpen

Abre un archivo de texto para lectura o escritura.

FileOpen (nombreFichero, [, mode])

Parámetros

nombreFichero	nombreFichero de texto para abrir.
[opcional] Modo modo para abrir el fichero como.	Modo (lectura o escritura) en la apertura del archivo. Puede ser una combinación de lo siguiente: 0 = Modo lectura (por defecto) 1 = Modo escritura (se coloca al final del fichero) 2 = Modo escritura (borra contenido previo)

	<p>8 = Crea una estructura de directorio si esta no existe(ver Comentarios).</p> <p>16 = Fuerza a modo binario (byte)(Ver Comentarios).</p> <p>32 = Usa modo Unicode UTF16 Little Endian en lectura y escritura. La lectura no sobreescribe el BOM existente.</p> <p>64 = Usa modo Unicode UTF16 Big Endian cuando escribe texto con <code>FileWrite</code> y <code>FileWriteLine</code> (por defecto usa ANSI)</p> <p>128 = Usa Unicode UTF8 (con BOM) modo lectura y escritura. La lectura no sobreescribe el BOM existente.</p> <p>256 = Usar Unicode UTF8 (sin BOM) modos lectura y escritura.</p> <p>16384 = Cuando se habre para lectura y no existe BOM, usa la detección completa de UTF8. Si esto no es usado entonces solo la parte inicial del fichero es chequeado para UTF8.</p> <p>La ruta del directorio debe existir siempre (excepto usando solo '8' - Ver Comentarios).</p>
--	---

Valor de Retorno

Con Éxitos Devuelve un identificador "handle" del archivo para uso con las subsecuentes funciones de fichero.

Al Fallar Devuelve -1 si ocurre un error.

Comentarios

- El identificador de fichero debe ser cerrado con la función [FileClose\(\)](#)
- Un fichero puede fallar al abrir debido a permisos o atributos.
- El modo por defecto cuando se escribe texto es ANSI - use el flag unicode para cambiar esto. Cuando se escriben ficheros unicode,el modo por defecto en Windows, (y más rápido en Autolt debido a la misma conversión) este es UTF16 Little Endian (modo 32).
- Abriendo un fichero en modo escritura se crea el fichero si este no existe. Los directorios no son creados a menos que se especifique el flag correcto.
- Cuando se lee o se escribe a traves de un identificador (handle) de fichero, la función [FileSetPos\(\)](#) debe ser usada para actualizar la posición actual dentro del fichero.

Relativo

[FileClose](#), [FileFlush](#), [FileRead](#), [FileReadLine](#), [FileWrite](#), [FileWriteLine](#), [FileGetPos](#), [FileSetPos](#)

Ejemplo

```
$file = FileOpen("test.txt", 0)
```

; Cheque si el fichero abierto es leido correctamente

```

If $file = -1 Then
    MsgBox(0, "Error", "Incapaz de abrir el fichero.")
    Exit
EndIf

FileClose($file)

```

; Otro ejemplo que crea automáticamente la estructura de directorio

```

$file = FileOpen("test.txt", 10) ; que es similar a 2 + 8 (borrar + crear directorio)
If $file = -1 Then
    MsgBox(0, "Error", "Incapaz de abrir el fichero.")
    Exit
EndIf
FileClose($file)

```

FileOpenDialog

Inicia un cuadro de diálogo de Abrir Archivo.

FileOpenDialog ("título", "init dir", "filtro" [, opciones [, "nombre por defecto" [, hwnd]]])

Parámetros

título	Texto título del Diálogo GUI.
init dir	Directorio inicial seleccionado en el árbol de archivos del GUI.
filtro	Filtro de ficheros simple tales como "All (*.*)" o "Text archivos (*.txt)" o filtro de ficheros múltiple tales como "All (*.*) Text archivos (*.txt)" (Ver Comentarios).
opciones	[opcional] Opciones de Diálogo: Para usar más de una opción, sumar los valores deseados. 1 = Fichero debe existir (si el usuario tipea un nombreFichero) 2 = Ruta debe existir (si el usuario tipea una ruta, finalizado con un backslash) 4 = Permite múltiple selección 8 = Prompt para crear un nuevo fichero (si este no existe)
nombre por defecto	[opcional] Sugerir un nombre de archivo para ser abierto. Por defecto esta vacío ("").
hwnd	[opcional] El identificador de ventana padre de este diálogo.

Valor de Retorno

Con Éxitos Devuelve la ruta completa del fichero(s) seleccionado. El resultado para múltiples selecciones se devuelve en la forma "Directory|fichero1|fichero2|..."

Al Fallar Establece @error a:

@error: 1 - falló la selección del fichero.

2 - filtro erróneo.

Comentarios

Separe filtros de múltiples archivos con un punto y coma ";".

Múltiples grupos de filtros están separados por un pipe "|".

Si el *nombre por defecto* es dado, Las opciones precedentes deben dadas también. Si no va utilizar las opciones, use 0 para estas.

Directarios especiales en Windows (tales como "My Documents") puede ser establecido como el *directorio inicial*; Ver apéndices.

@WorkingDir es cambiado cuando el valor de retorno es exitoso.

Relativo

[FileSaveDialog](#), [FileSelectFolder](#), [StringSplit](#)

Ejemplo

```
$message = "Mantenga oprimido Ctrl o Shift para escoger múltiples ficheros."
$var = FileOpenDialog($message, @WindowsDir & "\", "Images (*.jpg;*.bmp)", 1 + 4 )
If @error Then
  MsgBox(4096,"","No fichero(s) escogidos")
Else
  $var = StringReplace($var, "|", @CRLF)
  MsgBox(4096,"","Usted escogió " & $var)
EndIf
```

; Múltiples filtros de grupos

```
$message = "Mantenga oprimido Ctrl o Shift para escoger múltiples ficheros."
```

```
$var = FileOpenDialog($message, @WindowsDir & "", "Images (*.jpg;*.bmp)/Videos
(*.avi;*.mpg)", 1 + 4 )
```

```

If @error Then
    MsgBox(4096,"","No fichero(s) escogidos")
Else
    $var = StringReplace($var, "/", @CRLF)
    MsgBox(4096,"","Usted escogió " & $var)
Endif

```

FileRead

Lee un número de caracteres desde un fichero previamente abierto.

FileRead (IdentificadorFichero/NombreFichero [, contador])

Parámetros

IdentificadorFichero/NombreFichero	El identificador de un fichero, devuelto por una llamada previa a FileOpen. Alternativamente usted puede usar una cadena con el nombre del archivo como primer parámetro.
contador	[opcional] El número de caracteres para leer. Por defecto lee todo el fichero.

Valor de Retorno

Con Éxitos Devuelve la lectura binaria/cadena. @extended almacenará el número de bytes/caracteres devueltos.

Especial: Establece @error a -1 si el final del fichero es alcanzado.

Al Fallar Establece @error a 1 si archivo no se abre en modo lectura u otro error.

Comentarios

Si el nombre del archivo es dado en lugar del identificador - el archivo será abierto y cerrado durante la invocación de la función - para analizar ficheros de texto largos en mucho más lento de este modo que utilizando identificador de fichero.

Nota: Nunca mezcle identificador de fichero y nombre de fichero, ej., no utilice FileOpen para abrir un archivo y entonces use un nombre de archivo en esta función. Utilize cualquiera entre identificador de fichero y nombre de fichero en sus script, pero nunca ambos!

Ambos formatos de texto ANSI y UTF16/UTF8 pueden ser leídos - AutoIt determinará automáticamente el tipo.

Un archivo puede ser leído como dato binario(byte) usando FileOpen con el flag de binario- en este caso el contador es en bytes en vez de caracteres.

Relativo

[FileOpen](#), [FileReadLine](#), [FileWrite](#), [FileWriteLine](#), [String](#), [FileSetPos](#), [FileGetPos](#)

Ejemplo

```
$file = FileOpen("test.txt", 0)

; Cheque si el fichero abierto es leído correctamente
If $file = -1 Then
    MsgBox(0, "Error", "Incapaz de abrir el fichero.")
    Exit
EndIf

; lee un 1 por tiempo hasta que EOF (final del fichero) es alcanzado
While 1
    $chars = FileRead($file, 1)
    If @error = -1 Then ExitLoop
    MsgBox(0, "Caracter leído:", $chars)
Wend

FileClose($file)
```

FileReadLine

Lee una línea de texto desde un fichero previamente abierto.

FileReadLine (IdentificadorFichero/NombreFichero [, linea])

Parámetros

IdentificadorFichero/NombreFichero	El identificador de un fichero, devuelto por una llamada previa a FileOpen. Alternativamente usted puede usar una cadena con el nombre del archivo como primer parámetro.
------------------------------------	---

línea	[opcional] El número de la línea a leer. La primera línea en un archivo de texto es la línea 1 (no cero), la última línea es -1.
-------	---

Valor de Retorno

Con Éxitos Devuelve una línea de texto.

Especial: Establece @error a -1 si el final del fichero es alcanzado.

Al Fallar Establece @error a 1 si archivo no se abre en modo lectura u otro error.

Comentarios

Devuelve el texto de la línea leída, cualquier de los caracteres de nueva línea (CHR(10) o @LF) al final de una línea leída son automáticamente cortados.

Si el número de línea no es dado, la "próxima" línea será leída. ("Próxima" para un nuevo archivo abierto es la primera línea).

Si el nombre del archivo es dado en lugar del identificador - el archivo será abierto y cerrado durante la invocación de la función - para analizar ficheros de texto largos en mucho más lento de este modo que utilizando identificador de fichero.

Nota: Nunca mezcle identificador de fichero y nombre de fichero, ej., no utilice FileOpen para abrir un archivo y entonces use un nombre de archivo en esta función. Utilize cualquiera entre identificador de fichero y nombre de fichero en sus script, pero nunca ambos!

Desde el punto de vista del desempeño es una mala idea leer línea por línea especificando el parámetro "línea" con incremento a uno. Esto fuerza a Autolt a releer una y otra vez todo el archivo desde el comienzo hasta la línea especificada.

Ambos formatos de texto ANSI y UTF16/UTF8 pueden ser leídos - Autolt determinará automáticamente el tipo.

Relativo

[IniRead](#), [FileOpen](#), [FileRead](#), [FileWrite](#), [FileWriteLine](#), [FileSetPos](#), [FileGetPos](#)

Ejemplo

```
$file = FileOpen("test.txt", 0)
```

; Chequea si el fichero abierto es leído correctamente

```

If $file = -1 Then
    MsgBox(0, "Error", "Incapaz de abrir el fichero.")
    Exit
EndIf

; Lee las líneas de texto mientras no es encontrado EOF (final del fichero)
While 1
    $line = FileReadLine($file)
    If @error = -1 Then ExitLoop
    MsgBox(0, "Línea leída:", $line)
Wend

FileClose($file)

```

FileRecycle

Envía un archivo o directorio a la papelera de reciclaje (recycle bin).

FileRecycle ("fuente")

Parámetros

fuente	Dirección de un archivo(s) o directorio a reciclar. Comodines son soportados.
--------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 (Típicamente significa que el archivo está en uso o no existe).

Comentarios

Ver [FileFindFirstFile](#) para una discusión sobre los comodines.

Para remover un directorio, simplemente provea la ruta completa sin backslash al final.

Relativo

[FileDelete](#), [FileRecycleEmpty](#), [DirRemove](#), [FileMove](#)

Ejemplo

FileRecycle("C:.tmp")*

FileRecycleEmpty

Vacia la papelera de reciclaje.

FileRecycleEmpty (["recurso"])

Parámetros

recurso	[opcional] la ruta-raíz para vaciar - si es omitido este valor se vacía en todas las unidades.
---------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 (la papelera de reciclaje no pudo ser vaciada - ver más abajo).

Comentarios

Para que esta función trabaje IE4+ debe estar disponible.

Relativo

[FileDelete](#), [FileRecycle](#)

Ejemplo

FileRecycleEmpty("C:\")

FileSaveDialog

Inicializa un diálogo de Salvar Fichero.

FileSaveDialog ("título", "dir inicial", "filtro" [, opciones [, "nombre por defecto" [, hwnd]]])

Parámetros

título	Título del Diálogo GUI.
dir inicial	Directorio inicial mostrado en el árbol de la GUI.
filtro	Filtro simple de tipo de archivo como "Todos (*.*)" o "files de texto (*.txt)" o filtro múltiple de tipos de ficheros tales como "Todos (*.*) files de texto (*.txt)" (Ver Comentarios)
opciones	[opcional] 2 = La ruta debe existir (si el usuario tipea una ruta, esta debe terminar en backslash)
nombre por defecto	16 = Prompt para sobrescribir ficheros. [opcional] Nombre sugestivo de fichero para ser mostrado al inicio. Por defecto es blank ("").
hwnd	[opcional] El identificador de la ventana padre para este diálogo.

Valor de Retorno

Con Éxitos Devuelve la ruta completa del archivo escogido. Los resultados para múltiple selección son mostrado como : "Directory|file1|file2|..."

Al fallar Establece:

@error: 1 - Selección errónea del fichero.

2 - Filtro erróneo.

Comentarios

Separar los filtros de archivos por un ; como es mostrado en el ejemplo. La selección múltiple de archivos son separados por un pipe "|".

Si el nombre por defecto es dado, los parámetros previos deben ser dados también. Si ninguna de los parámetros son encontrados, utilice un 0.

Los directorios especiales de Windows (tales como "My Documents") pueden a veces ser establecido como el directorio de inicio; ver Apéndice.

@WorkingDir es cambiado cuando el valor de retorno es satisfactorio.

Relativo

[FileOpenDialog](#), [FileSelectFolder](#)

Ejemplo

```
$MyDocsFolder = ":::{450D8FBA-AD25-11D0-98A8-0800361B1103}"  
  
$var = FileSaveDialog( "Seleccionar un nombre.", $MyDocsFolder, "Scripts (*.aut;*.au3)", 2)  
; opción 2 = Diálogo permanente hasta validar la ruta/fichero seleccionado  
  
If @error Then  
    MsgBox(4096,"","Salvar cancelado.")  
Else  
    MsgBox(4096,"","Su selección " & $var)  
EndIf  
  
;Grupo de filtro múltiple  
$var = FileSaveDialog( "Seleccione los nombres.", $MyDocsFolder, "Scripts  
(*.aut;*.au3)|Ficheros texto (*.ini;*.txt)", 2)  
; opción 2 = Diálogo permanente hasta validar la ruta/fichero seleccionado  
  
If @error Then  
    MsgBox(4096,"","Salvar cancelado.")  
Else  
    MsgBox(4096,"","Su selección " & $var)  
EndIf
```

FileSelectFolder

Inicializa un diálogo de selección de carpeta.

FileSelectFolder ("texto diálogo ", "dir root" [, flag [, "dir inicial" [, hwnd]]])

Parámetros

texto diálogo	Texto de bienvenida del diálogo.
dir root	directorio Root del árbol de la GUI. Use "" para mostrar el Escritorio como root.
flag	[opcional] 1 = Muestra Botón Crear Carpeta (requiere IE6.0 o posterior) 2 = Usa el Diálogo de Nuevo Estilo (requiere IE5.0 o posterior) 4 = Muestra un Control Edit (para mostrar nombre de carpeta)
dir inicial	[opcional] Directorio Inicial/iniciar que será seleccionado si existe. Por defecto

	es blank ("").
hwnd	[opcional] El identificador de la ventana padre para este diálogo.

Valor de Retorno

Con Éxitos Devuelve la ruta completa de la carpeta escogida.

Al Fallar Devuelve "" (cadena vacía) y fija @error a 1 si usuario cancela/cierra la ventana.

Comentarios

El directorio root será establecido en caso de que el directorio inicial (si es dado) no exista. Un directorio root no existente provoca que se tome al Escritorio como directorio root.

La opción "Botón Crear Carpeta" puede requerir Windows XP con IE6 para que funcione. Los directorios especiales de Windows (Tales como "My Documents") pueden ser fijados como root usando el CLSID correcto según es mostrado en Apéndice.

Rutas UNC no son soportados. Si usted considera que los usuario pueda seleccionar ficheros en una ruta UNC entonces esa ruta necesita ser mapeada en una unidad primero.

Relativo

[FileSaveDialog](#), [FileOpenDialog](#)

Ejemplo

```
$var = FileSelectFolder("Seleccione una carpeta.", "")
```

FileSetAttrib

Establece los atributos de uno o más ficheros.

FileSetAttrib ("fichero", "+RASHNOT" [, recursión])

Parámetros

fichero	File(s) a cambiar, Eje. C:*.au3, C:\Dir
---------	--

+RASHNOT	Atributo(s) para fijar/borrar. Eje. "+A", "+RA-SH"
recursión	[opcional] Si es fijado a 1, entonces los directorios son recursivamente procesados. Por defecto es 0 (no recursión).

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si encuentra cualquier error.

Comentarios

El *fichero* no puede contener espacios!

Los atributos que pueden ser modificados con esta función con + o - son:

"R" = READONLY (solo lectura)
 "A" = ARCHIVE (Archivo)
 "S" = SYSTEM (Sistema)
 "H" = HIDDEN (Oculto)
 "N" = NORMAL (Normal)
 "O" = OFFLINE
 "T" = TEMPORARY (Temporal)

(Note que usted no puede fijar el atributo comprimido/directorio con esta función)

Si el parámetro recursión es fijado a 1 entonces todos los ficheros que se encuentren en los directorios hijos heredaran los mismos permisos.

Relativo

[FileGetAttrib](#), [FileGetTime](#), [FileSetTime](#)

Ejemplo

```
;marca todos los ficheros *.au3 en el actual directorio como solo lectura y sistema
If Not FileSetAttrib("*.au3", "+RS") Then
    MsgBox(4096, "Error", "Problema estableciendo atributos.")
EndIf
```

```
;marca todos los ficheros *.au3 en C:\ y subdirectorios como escribibles y archivo
If Not FileSetAttrib("C:\*.bmp", "-R+A", 1) Then
    MsgBox(4096, "Error", "Problema estableciendo atributos.")
```

EndIf

FileSetPos

Establece la posición actual en el fichero.

FileSetPos (identificador, desplazamiento, origen)

Parámetros

identificador	Un identificador a un fichero previamente abierto con FileOpen().
desplazamiento	El desplazamiento a efectuar desde la posición de origen. Este valor puede ser positivo o negativo. Valores negativos provocan desplazamiento hacia atrás desde el origen.
origen	Debe ser uno de los siguientes: 0 - Principio del fichero (\$FILE_BEGIN desde Constants.au3). 1 - Posición actual (\$FILE_CURRENT desde Constants.au3). 2 - Fin del fichero (\$FILE_END desde Constants.au3).

Valor de Retorno

Con Éxitos: True si la operación es satisfactoria

Al Fallar: False.

Comentarios

El fichero Include Constants.au3 debe ser colocado en su script para poder utilizar los nombres simbólicos entre paréntesis para establecer el parámetro origen.

Usando FileSetPos() es posible leer y escribir un fichero. Cuando se intente escribir y leer un fichero, siempre llame FileFlush() entre cada operación de escritura y lectura. Moviendo el puntero en medio de un dato puede ser usado para sobreescribir el dato.

Relativo

[FileGetPos](#), [FileFlush](#), [FileRead](#), [FileReadLine](#), [FileWrite](#), [FileWriteLine](#), [FileOpen](#)

Ejemplo

```
#include <Constants.au3>

Local Const $sFile = "test.txt"
Local $hFile = FileOpen($sFile, 2)

; Chequea si el fichero es correctamente abierto para escritura.
If $hFile = -1 Then
    MsgBox(0, "Error", "Unable to open file.")
    Exit
EndIf

; Escribiendo algo en el fichero.
FileWriteLine($hFile, "Line1")
FileWriteLine($hFile, "Line2")
FileWriteLine($hFile, "Line3")

; Flusheo del fichero en el disco.
FileFlush($hFile)

; Chequeando la posición en el fichero y probando leer el contenido de la posición actual.
MsgBox(0, "", "Position: " & FileGetPos($hFile) & @CRLF & "Data: " & @CRLF &
FileRead($hFile))

; Ahora, ajusta la posición al inicio del fichero.
Local $n = FileSetPos($hFile, 0, $FILE_BEGIN)

; Chequeando la posición del fichero y probando leer el contenido de la posición actual.
MsgBox(0, "", "Position: " & FileGetPos($hFile) & @CRLF & "Data: " & @CRLF &
FileRead($hFile))

; Cierra el identificador.
FileClose($hFile)

; Limpia el fichero temporal
FileDelete($sFile)
```

FileSetTime

Establece la marca de tiempo de uno o más ficheros

FileSetTime ("fichero", "tiempo" [, tipo [, recursión]])

Parámetros

fichero	Fichero(s) para cambiar, Eje. C:*.au3, C:\Dir
tiempo	El nuevo tiempo a fijar en el formato "YYYYMMDDHHMMSS" (Año, Mes, Día, Horas (Reloj 24hr), segundos). Si el tiempo es blank "" entonces la fecha y hora actual es usado.
tipo	[opcional] La marca de tiempo a cambiar: 0 = Modificado (por defecto), 1 = Creado, 2 = Accedido
recursión	[opcional] Si esto es fijado a 1, entonces los directorios son procesados recursivamente. Por defecto es 0 (no recursión).

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si ocurre un error cambiando la marca de tiempo(s).

Comentarios

Usando una fecha anterior a 1980-01-01 no tendrá efecto.

Tratando de establecer la marca de tiempo en una fichero de solo lectura este causará un error.

Relativo

[FileGetTime](#), [FileGetAttrib](#), [FileSetAttrib](#)

Ejemplo

;Cambia el fichero file.au3 "modificando" la marca de tiempo al 1ro de noviembre del 2003 y la hora actual

```
$var = FileSetTime("file.au3", "20031101")
```

FileWrite

Agrega un texto/dato al final de un fichero previamente abierto.

FileWrite ("filehandle/nombre de fichero", "texto/dato")

Parámetros

filehandle/nombre de fichero	El apuntador (handle) del fichero, devuelto por una llamada previa a FileOpen. Alternativamente, usted puede usar una ruta de archivo completa.
texto/dato	El texto/dato a escribir en el fichero. La línea de texto es escrita tal como es - sin caracteres @CR o @LF. Ver Comentarios para tipos de datos.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el archivo no puede abrirse en modo escritura, si es de solo lectura, o si el archivo no puede ser escrito de alguna forma.

Comentarios

El fichero de texto debe ser abierto en modo escritura o FileWrite fallará.

Si un nombre de archivo es dado en vez del identificador de archivo (handle), el archivo será abierto y cerrado durante la operación de escritura. Para analizar largos archivos de texto este método será mucho más lento que utilizando identificadores de archivo. Sin embargo, el nombre de archivo será creado si no existe.

Nota: No mezcle nombre de ficheros con identificadores de fichero, ej., no utilice FileOpen() para abrir el fichero de texto y utilice el nombre de fichero en esta función. Utilice cualquiera de los dos métodos pero no ambos a la vez.

Cuando se escribe en un texto Autolt escribirá utilizando ANSI por defecto. Para escritura en modo Unicode el archivo sebe ser abierto con FileOpen() y el flags adecuado.

Si el dato es un binario de tipo variant (y no es texto) este será escrito en el archivo byte por byte. Las operaciones con binario pueden también forzarse utilizando Fileopen con el flag binario.

Relativo

[FileFlush](#), [FileOpen](#), [FileRead](#), [File.ReadLine](#), [File.WriteLine](#), [Binary](#), [FileSetPos](#), [FileGetPos](#)

Ejemplo

```

$file = FileOpen("test.txt", 1); Chequea si el fichero abierto es escrito correctamente
If $file = -1 Then
    MsgBox(0, "Error", "Incapáz de abrir el fichero.")
    Exit
EndIf

FileWrite($file, "Línea1")
FileWrite($file, "Línea1 contigua" & @CRLF)
FileWrite($file, "Línea2")
FileClose($file)

```

FileWriteLine

Agrega una línea de texto al final de un fichero de texto previamente abierto.

FileWriteLine ("filehandle/nombre de fichero", "línea")

Parámetros

filehandle/nombre de fichero	El apuntador (handle) del fichero, devuelto por una llamada previa a FileOpen. Alternativamente, usted puede usar una ruta de archivo completa.
línea	La línea de texto a escribir en el fichero. Si la línea NO termina en @CR o @LF entonces un DOS linefeed (@CRLF) será automáticamente adicionado.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el archivo no puede abrirse en modo escritura, si es de solo lectura, o si el archivo no puede ser escrito de alguna forma.

Comentarios

El archivo de texto debe ser abierto en modo escritura o este FileWrite fallará.

Si un nombre de archivo es dado en vez del identificador de archivo(handle), el archivo será abierto y cerrado durante la operación de escritura. Para analizar largos archivos de texto este método será mucho más lento que utilizando identificadores de archivo. Sin embargo, el nombre de archivo será creado si no existe.

Nota: No mezcle nombre de ficheros e identificador de fichero, ej. no utilice FileOpen() para abrir el fichero de texto y utilice el nombre de fichero en esta función. Utilize cualquiera de los dos métodos pero no ambos a la vez.

Cuando se escribe en un texto AutoIt escribirá utilizando ANSI por defecto. Para escritura en modo Unicode el archivo se debe ser abierto con FileOpen() y el flags adecuado.

El texto escrito no puede contener caracteres Chr(0). La salida es truncada. FileWrite en modo binario puede ser utilizado para escribir como caracteres.

Relativo

[FileFlush](#), [FileOpen](#), [FileRead](#), [FileReadLine](#), [FileWrite](#), [FileSetPos](#), [FileGetPos](#)

Ejemplo

```
$file = FileOpen("test.txt", 1)

; Chequea si el fichero abierto es escrito correctamente
If $file = -1 Then
    MsgBox(0, "Error", "Incapáz de abrir el fichero.")
    Exit
EndIf

FileWriteLine($file, "Línea1")
FileWriteLine($file, "Línea2" & @CRLF)
FileWriteLine($file, "Línea3")

FileClose($file)
```

IniDelete

Borra un valor de un fichero .ini con formato estándar.

IniDelete ("nombre de fichero", "sección" [, "clave"])

Parámetros

nombre de fichero	El nombre de archivo del fichero .ini
sección	El nombre de la sección en del fichero .ini

clave	[opcional] El nombre de la clave en el fichero .ini para borrar. Si el nombre de la clave no es dado se borra la sección completa. La palabra reservada Default también puede ser usada para causar que la sección sea borrada.
-------	--

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el archivo INI no existe o si el archivo es de solo lectura.

Comentarios

Un archivo ini estándar tiene la forma:

[NombreDeSeccion]

Clave=Valor

Relativo

[IniRead](#), [IniWrite](#), [IniReadSection](#), [IniReadSectionNames](#), [IniRenameSection](#), [IniWriteSection](#)

Ejemplo

IniDelete("C:\Temp\mifile.ini", "sección2", "Clave")

IniRead

Lee un valor desde un fichero .ini con formato estándar.

IniRead ("nombre de fichero", "sección", "clave", "por defecto")

Parámetros

nombre de fichero	El nombre de archivo del fichero .ini
"sección"	El nombre de la sección del fichero .ini
clave	El nombre de clave en el en el .ini file.

por defecto	El valor predefinido de retorno si la clave dada no existe.
-------------	---

Valor de Retorno

Con Éxitos Devuelve el valor de la clave dada.

Al Fallar Devuelve la cadena *por defecto* si la clave solicitada no es encontrada.

Comentarios

Un archivo ini estándar tiene la forma:

[NombreDeSección]

Clave=Valor

Relativo

[IniDelete](#), [IniWrite](#), [FileReadLine](#), [IniReadSection](#), [IniReadSectionNames](#), [IniRenameSection](#), [IniWriteSection](#)

Ejemplo

```
$var = IniRead("C:\Temp\mifile.ini", "sección2", "Clave", "NoEncontrado")
MsgBox(4096, "Resultado", $var)
```

IniReadSection

Lee todos los pares de valores/claves de una sección en un fichero .ini con formato estándar.

IniReadSection ("nombre de fichero", "sección")

Parámetros

nombre de fichero	El nombre de archivo del fichero .ini
sección	El nombre de la sección en del fichero .ini

Valor de Retorno

Con Éxitos	Devuelve un arreglo de 2 dimensiones donde el elemento[n][0] es la clave y el elemento[n][1] es el valor.
Al Fallar	Establece @error=1 si es deshabilitado para leer la sección (El archivo INI no existe o la sección no existe)

Comentarios

Un archivo ini estándar tiene la forma:

```
[NombreDeSeccion]
Clave=Valor
```

El número de valores devueltos se guardan en \$result[0][0]. Si un @error ocurre, el arreglo no es creado.

Solo los primeros 32767 caracteres son tomados en cuenta en una sección para compatibilidad con Win9x.

Relativo

[IniDelete](#), [IniWrite](#), [IniRead](#), [IniReadSectionNames](#), [IniRenameSection](#), [IniWriteSection](#)

Ejemplo

```
$var = IniReadSection("C:\Temp\myfile.ini", "section2")
If @error Then
  MsgBox(4096, "", "Ocurrió un error, probablemente no es un archivo INI.")
Else
  For $i = 1 To $var[0][0]
    MsgBox(4096, "", "Clave: " & $var[$i][0] & @CRLF & "Valor: " & $var[$i][1])
  Next
EndIf
```

IniReadSectionNames

Lee todas las secciones en un fichero .ini con formato estándar.

IniReadSectionNames ("nombre de fichero")

Parámetros

nombre de	El nombre de archivo del fichero .ini
-----------	---------------------------------------

fichero	
---------	--

Valor de Retorno

Con Éxitos Devuelve un arreglo lineal con los nombres de todas las secciones del fichero INI.

Al Fallar Establece @error en fallo.

Comentarios

Un archivo ini estándar tiene la forma:

```
[NombreDeSeccion]
Clave=Valor
```

El número de elementos devueltos se guardan en \$result[0]. Si un @error ocurre, el arreglo no es creado.

Solo los primeros 32767 caracteres son tomados de una sección por compatibilidad con Win9x.

Relativo

[IniDelete](#), [IniWrite](#), [IniRead](#), [IniReadSection](#), [IniRenameSection](#), [IniWriteSection](#)

Ejemplo

```
$var = IniReadSectionNames(@WindowsDir & "\win.ini")
If @error Then
  MsgBox(4096, "", "Ocurrió un error, probablemente no es un archivo INI.")
Else
  For $i = 1 To $var[0]
    MsgBox(4096, "", $var[$i])
  Next
EndIf
```

IniRenameSection

Renombra una sección en un fichero .ini con formato estándar.

IniRenameSection ("nombre de fichero", "sección", "nueva sección" [, flag])

Parámetros

nombre de fichero	El nombre de archivo del fichero .ini.
sección	El nombre de la sección en del fichero .ini
nueva sección	El nuevo nombre de sección.
flag	[opcional] 0 (Por defecto) - Falla si "nueva sección" ya existe. 1 - Sobrescribe "nueva sección". Esto borrará cualquier clave existente en "nueva sección"

Valor de Retorno

Con Éxitos un entero distinto de cero

Al Fallar 0 y puede fijar @error si la sección no es sobrescrita (solo flag = 0).

Comentarios

Un archivo ini estándar tiene la forma:

[NombreDeSeccion]

Clave=Valor

Relativo

[IniRead](#), [IniDelete](#), [IniWrite](#), [IniReadSection](#), [IniReadSectionNames](#), [IniWriteSection](#)

Ejemplo

```
$res = IniRenameSection(@ScriptDir & "Mi.ini", "MiSección", "MiNuevaSección")
```

IniWrite

Escribe un valor para un fichero .ini con formato estándar.

IniWrite ("nombre de fichero", "sección", "clave", "valor")

Parámetros

nombre de fichero	El nombre de archivo del fichero .ini
sección	El nombre de la sección en del fichero .ini
clave	El nombre de la clave en del fichero .ini
valor	El valor para escribir/cambiar.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el archivo es de solo lectura.

Comentarios

Un archivo ini estándar tiene la forma:

*[NombreDeSeccion]
Clave=Valor*

Si el archivo no existe, este es creado. Cualquier directorio no existente es creado. Claves y/o secciones son adicionados al final y no son ordenados

Cuando se escribe un valor entre comillas, estas son quitadas. Si usted desea colocar un valor entre comillas entonces tiene que usarlas doble. Por ejemplo: ""Esto es una prueba"" lo cual produce "Esto es una prueba" en el fichero.

Espacios intermedios y finales son quitados. Si usted desea preservar el espacio, la cadena debe tener comillas. Por ejemplo: " Esto es una prueba" preservará el espacio, sin embargo, las comillas serán quitadas.

Valores en líneas múltiples no son soportados.

Relativo

[IniDelete](#), [IniRead](#), [IniReadSection](#), [IniReadSectionNames](#), [IniWriteSection](#), [IniRenameSection](#)

Ejemplo

```
IniWrite("C:\Temp\myfile.ini", "sección2", "Clave", "Esto es un nuevo valor")
```

IniWriteSection

Escribe una sección para un fichero .ini con formato estándar.

IniWriteSection ("nombre de fichero", "sección", "data" [, índice])

Parámetros

nombre de fichero	El nombre de archivo del fichero .ini
sección	El nombre de la sección en del fichero .ini
data	El dato a escribir. El dato puede ser una cadena o un arreglo. Si el dato es una cadena, entonces cada par clave=valor debe estar delimitado por @LF. Si el dato es un arreglo, el arreglo debe ser bidimensional y la segunda dimensión debe contener dos elementos.
índice	[opcional] Si un arreglo es pasado como dato, esto especifica el índice desde donde se iniciará la escritura. Por defecto, es 1 mientras que el valor devuelto por IniReadSection() puede ser usado inmediatamente. Para crear un arreglo manualmente, este valor necesita ser diferente en dependencia de como el arreglo ha sido creado. Este parámetro es ignorado si una cadena ha sido pasada como dato.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0. La función fija @error a 1 si el formato de dato es inválido.

Comentarios

Un archivo ini estándar tiene la forma:

```
[NombreDeSeccion]  
Clave=Valor
```

Si el archivo no existe, este es creado. Cualquier directorio no existente es creado. Claves y/o secciones son adicionados al final y no son ordenados.

Si la sección a escribir ya existe, el contenido será sobrescrito.

Relativo

[IniDelete](#), [IniRead](#), [IniReadSection](#), [IniReadSectionNames](#), [IniWrite](#), [IniRenameSection](#)

Ejemplo

; Éste es el archivo INI al que escribiremos. Será creado en el Desktop (escritorio).
\$sIni = @DesktopDir & "\AutoIt-Test.ini"

; Demuestra la creación de una nueva sección usando una cadena como entrada.

\$sData = "Key1=Valor1" & @LF & "Key2=Valor2" & @LF & "Key3=Valor3"

IniWriteSection(\$sIni, "Sección1", \$sData)

; Demuestra la creación de una nueva sección usando un arreglo como entrada.

\$aData1 = IniReadSection(\$sIni, "Sección1"); Leer lo que escribimos justamente arriba

For \$i = 1 To UBound(\$aData1) - 1

 \$aData1[\$i][1] &= "-" & \$i ; Cambia los datos un poco

Next

IniWriteSection(\$sIni, "Sección2", \$aData1); Escribe una nueva sección

; Demuestra la creación de un arreglo de forma manual y usarlo como entrada.

Dim \$aData2[3][2] = [["PrimeraClave", "PrimeraValor"], ["SegundaClave", "SegundoValor"],
["TerceraClave", "TercerValor"]]

; Puesto que el arreglo esta echo para iniciar en el elemento 0, nececitamos IniWriteSection()
para iniciar la escritura en el elemento 0.

IniWriteSection(\$sIni, "Sección3", \$aData2, 0)

Referencia de Funciones de Gráfico y Sonido

A continuación se muestra una lista completa de las funciones de Funciones de Gráfico y Sonido disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
Beep	Emite un sonido por el altavoz del sistema para el usuario.
PixelChecksum	Genera una suma de verificación(checksum) para una región de pixeles.
PixelCoordMode (Option)	establece la forma en que son utilizadas las coordenadas en funciones de píxel, ya sea para coordenadas absolutas o relativas a la ventana definida por hwnd [apuntador interno](por defecto la ventana activa):
PixelGetColor	Devuelve el color de un pixel de acuerdo a las coordenadas x,y del pixel.
PixelSearch	Busca un rectángulo de pixeles el color de pixel especificado.
SoundPlay	Ejecuta un archivo de sonido.
SoundSetWaveVolume	Establece el volumen de onda del sistema por porcentaje.

Beep

Emite un sonido por el altavoz del sistema para el usuario.

Beep ([frecuencia [, duración]])

Parámetros

frecuencia	[opcional] La frecuencia del sonido en hertz. Puede ser cualquiera de 37 hasta
------------	--

	32,767 (0x25 y 0x7FFF). Por defecto es 500 Hz.
duración	[opcional] La duración del sonido en milisegundos. Por defecto = 1000 MS.

Valor de Retorno

Siempre devuelve 1 indistintamente de la operación.

Comentarios

Ninguno

Relativo

[SoundPlay](#)

Ejemplo

*; emite un sólido , de frecuencia 500 por 1 segundo
Beep(500, 1000)*

PixelChecksum

Genera una suma de verificación (checksum), para una región de pixeles.

PixelChecksum (izquierda, superior, derecha, inferior [, paso [, hwnd [, modo]]])

Parámetros

izquierda	coordenada izquierda de rectángulo.
superior	coordenada superior de rectángulo.
derecha	coordenada derecha de rectángulo.
inferior	coordenada inferior de rectángulo.
paso	[opcional] En lugar de suma de verificación(checksum) de cada píxel usa un valor mayor que 1 para obviar pixeles (para velocidad). Ejem. Un valor de 2 solamente revisará cada 2 pixeles. Por defecto es 1. No es recomendado para

	usar un paso valor mayor que 1.
hwnd	[opcional] Apuntador de Ventana a ser usada.
modo	[opcional] Por defecto 0 ADLER checksum, 1 CRC32 Checksum.

Valor de Retorno

Con Éxito: Devuelve el valor checksum de la región.

Al Fallar: Devuelve 0.

Comentarios

La suma de verificación(checksum) solamente le permite ver si "algo" ha cambiado en una región - no le informa que ha cambiado exactamente.

Versiones previas fueron extremadamente lentas, sin embargo ahora la función es significativamente veloz. Usando el paso de parámetro no es más recomendado. El rendimiento ganado de usar un paso grande no es cercanamente como notable desde que la función ya es veloz con todo. También, un paso de mayor valor, es menos fiable en la suma de verificación (checksum) cuando viene a usarse para detectar pequeños cambios en una región.

El checksum CRC32 es mucho más lento que ADLDER pero detecta mejor las variaciones de pixel.

Relativo

[PixelGetColor](#), [PixelCoordMode \(Option\)](#), [PixelSearch](#)

Ejemplo

; Espera que algo cambie en la región 0,0 a 50,50

```
; Obtiene el checksum inicial
$checksum = PixelChecksum(0,0, 50,50)
```

; Espera a que la región cambie, la región es revisada cada 100ms para reducir la carga de CPU

```
While $checksum = PixelChecksum(0,0, 50, 50)
```

```
    Sleep(100)
```

```
    WEnd
```

MsgBox(0, "", "Algo en la región ha cambiado!")

PixelCoordMode

Establece la forma en que son utilizadas las coordenadas en funciones de píxel, ya sea para coordenadas absolutas o relativas a la ventana definida por hwnd [apuntador interno](por defecto la ventana activa):

- 0 = coordenadas relativas a la ventana actual.
- 1 =coordenadas absolutas a la pantalla (por defecto).
- 2 = coordenadas relativas a la ventana cliente de la ventana actual.

PixelGetColor

Devuelve el color de un pixel de acuerdo a las coordenadas x,y del pixel.

PixelGetColor (x , y [, hwnd])

Parámetros

x	coordenada x del pixel.
y	coordenada y del pixel.
hwnd	[opcional] Apuntador de Ventana a ser usado.

Valor de Retorno

Con Éxito: Devuelve **decimal** valor de pixeles color.

Al Fallar: Devuelve -1 si las coordenadas son inválidas.

Comentarios

Ninguno.

Relativo

[PixelSearch](#), [PixelCoordMode \(Option\)](#), [MouseGetPos](#), [PixelChecksum](#)

Ejemplo

```
$var = PixelGetColor( 10 , 100 )
```

```
MsgBox(0,"El color decimal es", $var)  
MsgBox(0,"El color hexadecimal es", Hex($var, 6))
```

PixelSearch

Busca un rectángulo de pixeles el color de pixel especificado.

PixelSearch (izquierda, superior, derecha, inferior, color [, variación-tono [, paso [, hwnd]]])

Parámetros

izquierda	coordenada izquierda del rectángulo.
superior	coordenada superior del rectángulo.
derecha	coordenada derecha del rectángulo.
inferior	coordenada inferior del rectángulo.
color	Color de pixel a buscar (en decimal o hexadecimal)
variación-tono	[opcional] Un número entre 0 y 255 para indicar el número permitido de tonos de variación de rojo, verde, y azul con los componentes del color. Por defecto es 0 (coincidencia exacta)
paso	[opcional] En lugar de buscar cada pixel usar un valor mayor que 1 para obviar pixeles (para velocidad). Ejem. A valor de 2 solamente revisará cada 2 pixeles. Por defecto es 1.
hwnd	[opcional] Apuntador de Ventana para ser usado.

Valor de Retorno

Con Éxito: Devuelve a dos-elementos arreglo de coordenadas del pixel. (Arreglo[0] = x, Arreglo[1] = y).

Al Fallar: Establece @error a 1 si el color no es encontrado.

Comentarios

La dirección de búsqueda se realiza como sigue:

Izquierda a derecha - izquierda < derecha

Derecha a izquierda - derecha < izquierda

Arriba a abajo - arriba < abajo

Abajo a arriba -abajo < arriba

Cambiar la dirección de búsqueda resulta muy útil si el color a buscar aparece en un cuadrante específico del área de búsqueda.

Recuerde, una típica pantalla normal a 1024 x 768 tiene 786432 pixeles. Aunque PixelSearch es optimizado, reduciendo el área de búsqueda ayuda a mejorar la velocidad del resultado.

Relativo

[PixelChecksum](#), [PixelGetColor](#), [PixelCoordMode \(Option\)](#)

Ejemplo

```
; Busca el pixel de rojo puro en el rango 0,0-20,300  
$coord = PixelSearch( 0, 0, 20, 300, 0xFF0000 )  
If Not @error Then  
    MsgBox(0, "X y Y son:", $coord[0] & "," & $coord[1])  
EndIf
```

```
; Busca el pixel de rojo puro dentro de 10 variaciones de rojo puro  
$coord = PixelSearch( 0, 0, 20, 300, 0xFF0000, 10 )  
If Not @error Then  
    MsgBox(0, "X y Y son:", $coord[0] & "," & $coord[1])  
EndIf
```

SoundPlay

Ejecuta un archivo de sonido.

SoundPlay ("nombre de archivo" [, esperar])

Parámetros

nombre de archivo	Nombre del archivo a ser ejecutado (típicamente un archivo WAV o MP3)
-------------------	---

esperar	<p>[opcional] Este flag determina si el script deberá esperar que el sonido finalice para continuar o que continúe el script:</p> <p>1 = esperar hasta que el sonido haya finalizado 0 = continua el script mientras el sonido está ejecutándose (predeterminado)</p>
---------	--

Valor de Retorno

Ninguno. (Siempre devuelve 1 independientemente de su éxito)

Comentarios

Terminando el script detendrá el sonido (si aún estuviese sonando).

Llamar *SoundPlay("")* puede ser usado para detener un sonido que actualmente está siendo escuchado. Esto también tiene el efecto de cerrar el apuntador del sonido abierto.

Si usted necesita eliminar un archivo de sonido el cual se está ejecutando en el script, usted debería llamar primero *SoundPlay("")* para asegurarse que el apuntador del sonido esté cerrado.

Relativo

[SoundSetWaveVolume](#), [Beep](#)

Ejemplo

SoundPlay(@WindowsDir & "\media\tada.wav",1)

SoundSetWaveVolume

Establece el volumen de onda del sistema por porcentaje.

SoundSetWaveVolume (por ciento)

Parámetros

porcentaje	porcentaje entre 0 y 100
------------	--------------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 y establece el valor de @error en 1 si el por ciento es inválido.

Comentarios

Este controla el volumen de Onda, no el volumen maestro. Un valor de cero no enmudece el audio.

En Windows Vista, no existe volumen de onda. Esta función solo cambia el volumen maestro para el script. Esto puede no ser usado para cambiar el volumen maestro de otros programas.

Relativo

[SoundPlay](#)

Ejemplo

SoundSetWaveVolume(50)

Referencia de Administración de GUI

A continuación se muestra una lista completa de las funciones de Administración de GUI disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
GUICtrlCreateAvi	Crea un control "avi de video" en el GUI.
GUICtrlCreateButton	Crea un control Botón en el GUI.
GUICtrlCreateCheckbox	Crea un control de caja de chequeo (Checkbox) en el GUI.
GUICtrlCreateCombo	Crea un control ComboBox en el GUI.
GUICtrlCreateContextMenu	Crea un menú contextual para un control o una ventana GUI entera.
GUICtrlCreateDate	Crea un control de fecha en la GUI.
GUICtrlCreateDummy	Crea un control Dummy en la GUI.
GUICtrlCreateEdit	Crea un control Edit en la GUI.
GUICtrlCreateGraphic	Crea un control Graphic en la GUI.
GUICtrlCreateGroup	Crea un control de Grupo en la GUI.
GUICtrlCreateIcon	Crea un control Icono en la GUI.
GUICtrlCreateInput	Crea un control Input en la GUI.
GUICtrlCreateLabel	Crea un control de etiqueta estática en la GUI.
GUICtrlCreateList	Crea un control List en la GUI.
GUICtrlCreateListView	Crea un control ListView en la GUI.
GUICtrlCreateListViewItem	Crea un item en un control ListView.
GUICtrlCreateMenu	Crea un control Menu en la GUI.
GUICtrlCreateMenuItem	Crea un control MenuItem en la GUI.
GUICtrlCreateMonthCal	Crea un control de calendario para el GUI.

<u>GUICtrlCreateObj</u>	Crea un control ActiveX en la GUI.
<u>GUICtrlCreatePic</u>	Crea un control de Imagen para el GUI.
<u>GUICtrlCreateProgress</u>	Crea un control de barra de progreso en la GUI.
<u>GUICtrlCreateRadio</u>	Crea un control botón de Radio en la GUI.
<u>GUICtrlCreateSlider</u>	Crea un control Slider en la GUI.
<u>GUICtrlCreateTab</u>	Crea un control de Tab en la GUI.
<u>GUICtrlCreateTabItem</u>	Crea un control TabItem en la GUI.
<u>GUICtrlCreateTreeView</u>	Crea un control TreeView en la GUI.
<u>GUICtrlCreateTreeViewItem</u>	Crea un control TreeViewItem en la GUI.
<u>GUICtrlCreateUpdown</u>	Crea un control UpDown en la GUI.
<u>GUICtrlDelete</u>	Elimina un control.

Referencia de Administración del Teclado

A continuación se muestra una lista completa de las funciones de Administración del Teclado disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
HotKeySet	Establece el método de teclado para una función de usuario determinada.
Send	Envía pulsaciones simuladas de teclas a la ventana activa.
SendAttachMode (Option)	Especifica si AutoIt adjunta las entradas cuando es usada la función Send(). Cuando no están adjuntas (por defecto=0) detectando el estado de capslock/scrolllock y numlock pueda ser poco confiables en Windows NT4. Sin embargo, cuando específicas y adjunta modo=1 la sintaxis Send("{... down/up}") no trabajará y haya tal vez problemas enviando teclas a una ventana "colgada". ControlSend() SIEMPRE adjunta y no es afectada por este modo.
SendCapslockMode (Option)	Especifica si AutoIt debería almacenar el estado de capslock antes de una función Send y lo restaura posteriormente.
SendKeepActive	Intenta mantener activa una ventana específica durante Send().
SendKeyDelay (Option)	Altera la duración de brevedad entre envíos de teclas. Un valor de 0 remueve la espera completamente.
SendKeyDownDelay (Option)	Altera la duración de tiempo entre el momento que una tecla está presionada antes de liberarla. Para aplicaciones es necesario registrar las teclas presionadas (y en muchos juegos) tal vez necesite aumentar este valor del que viene por defecto. Un valor de 0 remueve la espera completamente.

HotKeySet

Establece el método de teclado para una función de usuario determinada.

HotKeySet ("tecla" [, "función"])

Parámetros

tecla	La combinación de tecla utilizado en el método de teclado. algún formato como en Send().
función	[opcional] El nombre de la función a invocar cuando la tecla es presionada. Si no se especifica este parámetro el método de teclado definido se deshabilita.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

Si dos script de Autoit utilizan el mismo método de teclado, usted debería evitar correr ambos scripts simultáneamente. (El segundo script ejecutado no podrá capturar el método de teclado a menos que el primer script termine o desabilite el método de teclado alterando el segundo parámetro de la función: poniéndolo a "")

Cuando se activa un método de teclado "típicamente" el script de Autoit se interrumpe, e invoca a la función de usuario, hasta que esta es completada o interrumpida. Las excepciones son las siguientes:

- 1) Si la función actual es una función de "bloqueo", entonces el método de teclado es detenido y no es ejecutado hasta tanto la función de bloqueo finalice. MsgBox y FileSelectFolder son ejemplo de funciones de bloqueo. Pruebe el comportamiento de Shift-Alt-d en el Ejemplo.
- 2) Si usted tiene el script pausado por un evento clic sobre el icono de la barra de tareas, cualquier método de teclado invocado durante esta pausa es ignorado.

Los siguientes métodos de teclados no pueden ser establecidos:

Ctrl+Alt+Delete	Este es reservado por Windows.
F12	Este es también reservado por Windows, en correspondencia con una API.
NumPad's Enter Key	En su lugar, use {Enter} que captura ambas teclas Enter del teclado.
Win+B,D,E,F,L,M,R,U; y Win+Shift+M	Esto es un acceso directo intrínseco de Windows. Nota: Win+B y Win+L pueden ser reservados solamente para Windows XP y superior.
Alt, Ctrl, Shift, Win	Estas se reservan de manera intrínseca!
Otros	Cualquier otro método de teclado general es ajustado por vía Software, cualquier conjunto de dos o más "teclas de base" tales como

	{F1}{F2}', y cualquier tecla de la forma '{LALT}' o '{ALTDOWN}'.
--	--

Cuando usted establece un método de teclado, AutoIt captura las teclas presionadas y la pasa a la aplicación activa, con una excepción: Las teclas de bloqueo (NumLock, CapsLock, y ScrollLock) Todas actúan según su condición respectiva!

Para la función Send() una combinación de teclas desencadena un evento HotKeySet(), puede usar cualquiera de los dos: ControlSend() o inhabilitar el evento HotKeySet(), de otro modo, el evento Send() puede desencadenar un ciclo infinito.

```
; captura y pasa las teclas presionadas
HotKeySet("{Esc}", "captureEsc")
Func captureEsc()
; ... puede colocar código aquí
HotKeySet("{Esc}")
Send("{Esc}")
HotKeySet("{Esc}", "captureEsc")
EndFunc
```

La función llamada puede **NO** tener parámetros. Estos serán ignorados.

La macro @HotKeyPressed puede ser utilizado dentro de la función para indicar varias teclas en la misma función.

Relativo

[Send](#), [GUISetAccelerators](#)

Ejemplo

; Presione Esc para terminar el script, Pausa/interrumpir para "pausar"

```
Global $Paused
HotKeySet("{PAUSE}", "TogglePause")
HotKeySet("{ESC}", "Terminate")
HotKeySet("+!d", "ShowMessage") ;Shift-Alt-d
```

;;;; Cuerpo del programa va aquí ;;;

```
While 1
Sleep(100)
WEnd
;;;;;
```

```
Func TogglePause()
$Paused = NOT $Paused
While $Paused
```

```

sleep(100)
ToolTip('Script es "Pausado"',0,0)
WEnd
ToolTip("")
EndFunc

Func Terminate()
Exit 0
EndFunc

Func ShowMessage()
MsgBox(4096,"","Esto es un mensaje.")
EndFunc

```

Send

Envía pulsaciones simuladas de teclas a la ventana activa.

Send ("teclas" [, bandera])

Parámetros

teclas	La secuencia de teclas a enviar.
bandera	[opcional] Cambia como las "teclas" son procesadas: bandera = 0 (por defecto), Texto contiene caracteres especiales como + y ! para indicar las teclas de SHIFT y ALT presionadas. bandera = 1, teclas son enviadas en modo sin procesar (raw)

Valor de Retorno

Ninguno.

Comentarios

Ver el [Apéndice](#) para algunos consejos usando Send. Autolt puede enviar todo los caracteres ASCII y ASCII Extendido (0-255), para enviar caracteres UNICODE se debe usar la opción "ASC" y el código del carácter que se desea enviar (ver {ASC} a final de la tabla de abajo).

La sintaxis del comando "Send" es similar a la de ScriptIt y en Visual Basic el comando "SendKeys". Los caracteres son enviados como son escritos con la excepción de los siguientes caracteres:

'!

Este le dice a Autolt que envíe una tecla ALT, por tanto Send("Este es un texto!a") enviaría la teclas "Este es un texto" y luego presionaría "ALT+a".

Nota: Algunos programas son muy selectivos sobre letras mayúsculas y teclas ALT, por ej. "!A" es diferente a "!a". El primero dice ALT+SHIFT+A, el segundo es ALT+a. En caso de duda, utilice minúsculas!

'+'

Este le dice a Autolt que envíe la tecla SHIFT, por tanto Send("Hello") enviaría el texto "Hello". Send("!+a") enviaría "ALT+SHIFT+a".

'^'

Este le dice a Autolt que envía la tecla CONTROL, por tanto Send("^!a") enviaría "CTRL+ALT+a".

Nota: Algunos programas son muy selectivos sobre letras mayúsculas y teclas CTRL, por ej. "^A" es diferente a "^a". El primero dice CTRL+SHIFT+A, el segundo es CTRL+a. En caso de duda, utilice minúsculas!

'#'

La representación del signo almohadilla o numeral envía la tecla de Windows(logo de Windows); por tanto, Send("#r") enviaría Win+r que abre la caja de diálogo de "Ejecutar" o su equivalente en inglés, "Run".

Usted puede establecer SendCapslockMode para desactivar CAPS LOCK al iniciar una operación Send y restaurarlo al completar la operación.

Sin embargo, si un usuario presiona la tecla Shift cuando una función de Send inicia, el texto puede ser enviado en mayúscula.

Una solución es enviar a Send("{SHIFTDOWN}{SHIFTUP}") antes que otras operaciones de Send.

Ciertos teclados como el Checo envía los caracteres de forma diferente cuando usando la tecla Shift o siendo que CAPS LOCK esté activo y enviando un carácter. Dado que la implementación de envío de Autolt, el CAPS LOCK será enviado como que si estuviese activo, así que SHIFT **no trabajará**.

Ciertas teclas especiales pueden ser enviadas y deberían ser encerradas en llaves ("{" y "}"):

Nota: Windows no permite la simulación de la combinación de "CTRL-ALT-DEL"!

Comando Send(si la bandera está en cero)	Teclas Presionadas
{!}	!

{#}	#
{+}	+
{^}	^
{}	{
}	}
{SPACE}	SPACE
{ENTER}	ENTER tecla principal en el teclado
{ALT}	ALT
{BACKSPACE} o {BS}	BACKSPACE
{DELETE} o {DEL}	DELETE
{UP}	Flecha Arriba
{DOWN}	Flecha Abajo
{LEFT}	Flecha Izquierda
{RIGHT}	Flecha Derecha
{HOME}	HOME
{END}	END
{ESCAPE} o {ESC}	ESCAPE
{INSERT} o {INS}	INS
{PGUP}	RePág (PageUp)
{PGDN}	AvPág (PageDown)
{F1} - {F12}	Teclas de Funciones
{TAB}	TAB

{PRINTSCREEN}	Tecla Impr pant (Print Screen)
{LWIN}	Tecla Izquierda de Windows
{RWIN}	Tecla Derecha de Windows
{NUMLOCK on}	NUMLOCK (on/off/toggle)
{CAPSLOCK off}	CAPSLOCK (on/off/toggle)
{SCROLLLOCK toggle}	SCROLLLOCK (on/off/toggle)
{BREAK}	para procesar Ctrl+Break
{PAUSE}	PAUSE
{NUMPAD0} - {NUMPAD9}	Dígitos del Numpad
{NUMPADMULT}	Multiplicador del Numpad
{NUMPADADD}	Numpad Agregar
{NUMPADSUB}	Numpad Sustraer
{NUMPADDIV}	Numpad Dividir
{NUMPADDOT}	Numpad punto
{NUMPADENTER}	Tecla Enter en el numpad
{APPSKEY}	Tecla de Aplicaciones de Windows
{LALT}	Tecla Izquierda de ALT
{RALT}	Tecla Derecha de ALT
{LCTRL}	Tecla Izquierda de CTRL
{RCTRL}	Tecla Derecha de CTRL
{LSHIFT}	Tecla Izquierda de Shift
{RSHIFT}	Tecla Derecha de Shift

{SLEEP}	Tecla SLEEP
{ALTDOWN}	Mantiene abajo la tecla ALT hasta que {ALTUP} es enviado
{SHIFTDOWN}	Mantiene abajo la tecla de SHIFT hasta que {SHIFTUP} es enviado
{CTRLDOWN}	Mantiene abajo la tecla de CTRL hasta que {CTRLUP} es enviado
{LWINDOW}	Mantiene abajo la tecla izquierda de Windows hasta que {LWINUP} es enviado
{RWINDOW}	Mantiene abajo la tecla derecha de Windows hasta que {RWINUP} es enviado
{ASC nnnn}	Envía la combinación de tecla de ALT+nnnn
{BROWSER_BACK}	2000/XP Solamente: Selecciona en el navegador la tecla de "retroceder"
{BROWSER_FORWARD}	2000/XP Solamente: Selecciona en el navegador la tecla de "adelantar"
{BROWSER_REFRESH}	2000/XP Solamente: Selecciona en el navegador la tecla de "refrescar"
{BROWSER_STOP}	2000/XP Solamente: Selecciona en el navegador la tecla de "detener"
{BROWSER_SEARCH}	2000/XP Solamente: Selecciona en el navegador la tecla de "buscar"
{BROWSER_FAVORITES}	2000/XP Solamente: Selecciona en el navegador la tecla de "favoritos"
{BROWSER_HOME}	2000/XP Solamente: Abre el navegador y se dirige a la página inicial o home
{VOLUME_MUTE}	2000/XP Solamente: Enmudece el volumen general
{VOLUME_DOWN}	2000/XP Solamente: Reduce el volumen

{VOLUME_UP}	2000/XP Solamente: Incrementa el volumen
{MEDIA_NEXT}	2000/XP Solamente: Selecciona la próxima pista en el reproductor de medios
{MEDIA_PREV}	2000/XP Solamente: Selecciona la pista anterior en el reproductor de medios
{MEDIA_STOP}	2000/XP Solamente: Detiene la ejecución del reproductor de medios
{MEDIA_PLAY_PAUSE}	2000/XP Solamente: Ejecuta/pausa el reproductor de medios
{LAUNCH_MAIL}	2000/XP Solamente: Ejecuta la aplicación de correo
{LAUNCH_MEDIA}	2000/XP Solamente: Ejecuta el reproductor de medios
{LAUNCH_APP1}	2000/XP Solamente: Ejecuta usuario app1
{LAUNCH_APP2}	2000/XP Solamente: Ejecuta usuario app2

Para enviar el valor ASCII de A (igual que presionar ALT+065 en el teclado numérico)

Send("{ASC 065}")

(Cuando son usados 2 dígitos de código ASCII debe insertarse un 0, de otro modo un código obsoleto 437 es usado).

Para enviar los caracteres UNICODE introducir el código del carácter (decimal o hex), por ejemplo este envía un carácter chino

Send("{ASC 2709}") o Send("{ASC 0xA95}")

Teclas sencillas pueden ser repetidas , ejemplo:

Send("{DEL 4}") ;Presiona la tecla DEL (Eliminar) 4 veces

Send("{S 30}") ;Envía 30 caracteres 'S'

Send("+{TAB 4}") ;Presiona la combinación SHIFT+TAB 4 veces

El tecla será enviada al menos una vez si el contador es cero.

Para mantener una tecla presionada hacia abajo (por lo general es útil solamente para juegos)

Send("{a down}") ;Presiona la tecla A

Send("{a up}") ;Libera la tecla A

Para establecer el estado de las teclas capslock, numlock y scrolllock

```

Send("{NumLock on}") ;Activa la tecla NumLock
Send("{CapsLock off}") ;Desactiva la tecla CapsLock
Send("{ScrollLock toggle}") ;Altera el estado de ScrollLock

```

Si usted desea usar una variable para el contador, intentar

```

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

```

Si usted desea enviar el valor ASCII de A cuatro veces, entonces intente

```

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

```

Muchas computadoras portátiles tienen una tecla especial Fn. Esta tecla no puede ser simulada.

Nota, al colocar la bandera parámetro a 1 por encima de "especial" de procesamiento será desactivado. Este es útil cuando usted desea enviar algunos textos copiados de una variable y usted quiere enviar el texto exactamente como fue escrito.

Para ejemplo, abrir Opciones de Carpeta (en el Panel de Control) e intentar lo siguiente:

Send("{TAB}")	Navega el próximo control (botón, checkbox, etc)
Send("+{TAB}")	Navega el control previo.
Send("^{TAB}")	Navega a través de las pestañas
Send("^+{TAB}")	Navega a través de las pestañas anteriores.
Send("{SPACE}")	Puede ser usado para alternar un checkbox o pulsar un botón.
Send("{+}")	Usualmente marca un checkbox (si es un checkbox "real".)
Send("{-}")	Usualmente desmarca un checkbox.
Send("{NumPadMult}")	Recursivamente expande carpetas en un SysTreeView32.

Use las combinaciones de tecla Alt para acceder a los ítems de menús. También, puede abrir Notepad e intentar lo siguiente:

Send("!a") Send Alt+a, al acceder vía tecla al menú de "Archivo". Intente otras letras!

Send("{DOWN}")	Despliega hacia abajo el menú.
----------------	--------------------------------

Send("{UP}")	Despliega hacia arriba dentro de un menú.
Send("{LEFT}")	Se desplaza hacia la izquierda o expande un submenú.
Send("{RIGHT}")	Se desplaza hacia la derecha a un nuevo menú o colapsa un submenú.

Para ver la ayuda de Windows presionar Win+F1--para una lista completa de accesos de teclado si usted no conoce la importancia de Alt+F4, PrintScreen, Ctrl+C, y otros.

Cuando se está ejecutando un script en una computadora remota a través de un programa como psexec (www.sysinternals.com) o beyondexec (www.beyondlogic.org) es necesario, especialmente cuando se envían teclados a un programa Ejecuta por el script con una función Run, para usar **ControlSend** u otras funciones de ControlXXX para comunicar directamente con el control. Aun enviando con Opt("SendAttachMode",1) no trabaja.

Usar el modo -s para cuando se quiere tener mejores derechos en la computadora remota.

Opt("SendKeyDelay",...) altera la longitud de la breve pausa entre teclas enviadas. Opt("SendKeyDownDelay",...) altera la longitud de tiempo a una tecla presionada antes de que sea liberada.

Establezca "SendKeyDelay" y "SendKeyDownDelay" a 0 para remover todas las esperas en los envíos de teclas. Esto puede ser requerido bajo ciertas circunstancias, por ejemplo, cuando abra el sistema ("#!") esto puede ser necesario para remover la espera en orden para prevenir que la tecla WIN sea oprimida.

Relativo

[SendAttachMode \(Option\)](#), [SendKeepActive](#), [SendKeyDelay \(Option\)](#), [SendKeyDownDelay \(Option\)](#), [ControlSend](#), [BlockInput](#), [HotKeySet](#), [WinMenuSelectItem](#)

Ejemplo

```
Send("#r")
WinWaitActive("Run")
Send("notepad.exe{Enter}")
WinWaitActive("[CLASS:Notepad]")
Send("La fecha/hora es {F5}")
```

SendAttachMode

Especifica si AutoIt adjunta las entradas cuando es usada la función Send(). Cuando no están adjuntas (por defecto=0) detectando el estado de capslock/scrolllock y numlock pueda ser poco confiables en Windows NT4. Sin embargo, cuando especificas y adjunta

modo=1 la sintaxis Send("{... down/up}") no trabajará y haya tal vez problemas enviando teclas a una ventana "colgada". ControlSend() SIEMPRE adjunta y no es afectada por este modo.

0 = no adjunta (por defecto)

1 = adjunta

SendCapslockMode

Especifica si AutoIt debería almacenar el estado de capslock antes de una función Send y lo restaura posteriormente.

0 = no almacena/restaura

1 = almacena y restaura (por defecto)

SendKeepActive

Intenta mantener activa una ventana específica durante Send().

SendKeepActive ("título" [, "texto"])

Parámetros

título	El título de la ventana a activar. Ver definiciones especiales de títulos . Usar un título en blanco para desactivar la función.
texto	[opcional] El texto de la ventana.

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la ventana no es encontrada.

Comentarios

Intenta reiniciar la ventana activa entre cada tecla simulada de Send().

Relativo

[Send](#)

Ejemplo

Run("notepad.exe")

```
WinWait("[CLASS:Notepad]")
SendKeepActive("[CLASS:Notepad]")
; Cambia la ventana activa mientras pausa
For $i = 1 to 10
  Sleep(1000)
  Send("Hola")
Next
```

SendKeyDelay

Altera la duración de brevedad entre envíos de teclas. Un valor de 0 remueve la espera completamente.

SendKeyDownDelay

Altera la duración de tiempo entre el momento que una tecla está presionada antes de liberarla. Para aplicaciones es necesario registrar las teclas presionadas (y en muchos juegos) tal vez necesite aumentar este valor del que viene por defecto. Un valor de 0 remueve la espera completamente.

FUNCIONES MATEMÁTICAS

A continuación se muestra una lista completa de las funciones de Funciones Matemáticas disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
Abs	Calcula el valor absoluto de un número.
ACos	Calcula el arco coseno de un número.
ASin	Calcula el arco seno de un número.
ATan	Calcula el arco tangente de un número.
BitAND	Realiza una operación AND a nivel de bit.
BitNOT	Realiza una operación NOT a nivel de bit.
BitOR	Realiza una operación OR a nivel de bit.
BitRotate	Realiza una operación de alternación de bit, con rotación.
BitShift	Realiza una operación de alternación de bit.
BitXOR	Realiza una operación OR exclusiva (XOR) a nivel de bit.
Cos	Calcula el coseno de un número.
Ceiling	Devuelve un número redondeado por encima del próximo entero.
Exp	Calcula la potencia e de un número.
Floor	Devuelve un número por debajo del entero más próximo.
Log	Calcula el logaritmo natural de un número.

<u>Mod</u>	Ejecuta una operación de módulo.
<u>Random</u>	Genera un número seudo-aleatorio del tipo flotante.
<u>Round</u>	Devuelve un número redondeado a un número determinado de cifras decimales.
<u>Sin</u>	Calcula el seno de un número.
<u>Sqrt</u>	Calcula la raíz cuadrada de un número.
<u>SRandom</u>	Establece un semilla para la generación de números aleatorios.
<u>Tan</u>	Calcula la tangente de un número.

Abs

Calcula el valor absoluto de un número.

Abs (expresión)

Parámetros

expresión	Cualquier expresión numérica válida.
-----------	--------------------------------------

Valor de Retorno

Devuelve el valor absoluto de *expresión*.

Comentarios

Una cadena puede tener valor cero.

Relativo

Ninguno.

Ejemplo

```
$var = Abs(-123.45) ;valor 123.45
```

ACos

Calcula el arco coseno de un número.

ACos (expresión)

Parámetros

expresión	Cualquier valor entre -1 y 1 inclusive.
-----------	---

Valor de Retorno

Devuelve el arco coseno trigonométrico de *expresión*. Resultado en radianes.

Comentarios

ACos(x) es matemáticamente definido entre $-1 \leq x \leq 1$, por eso ACos tiende a devolver -1.#IND para otros valores de x.

Relativo

[Sin](#), [Cos](#), [Tan](#), [ASin](#), [ATan](#)

Ejemplo

$$\$x = ACos(0.5)$$

$$\$pi = 3.14159265358979$$

$$\$radToDeg = 180 / \$pi$$

$\$y = ACos(-1) * \$radToDeg$;arcCosine de -1 retorna 180°

ASin

Calcula el arco seno de un número.

ASin (expresión)

Parámetros

expresión	Cualquier valor entre -1 y 1 (inclusive).
-----------	---

Valor de Retorno

Devuelve el arco seno trigonométrico de *expresión*. Resultados en radianes.

Comentarios

ASin(x) es matemáticamente definido para valores entre $-1 \leq x \leq 1$, por eso ASin tiende a devolver -1.#IND para otros valores de x.

Relativo

[Sin](#), [Cos](#), [Tan](#), [ACos](#), [ATan](#)

Ejemplo

$\$x = ASin(0.5)$

$\$pi = 3.14159265358979$

$\$radToDeg = 180 / \pi

$\$y = ASin(1) * \$radToDeg$; arcsine de 1 retorna 90°

ATan

Calcula el arco tangente de un número.

ATan (expresión)

Parámetros

expresión	Cualquier expresión numérica válida.
-----------	--------------------------------------

Valor de Retorno

Devuelve el arco tangente trigonométrico de *expresión*. Resultados en radianes.

Comentarios

El resultado de **4 * ATan(1)** es pi.

Relativo

[Sin](#), [Cos](#), [Tan](#), [ASin](#), [ACos](#)

Ejemplo

$\$x = ATan(0.5)$

$\$pi = 4 * ATan(1)$; igual a 3.14159265358979

$\$radToDeg = 180 / \pi

$\$y = ATan(1) * \$radToDeg$; arctangent de 1 retorna 45°

BitAND

Realiza una operación AND a nivel de bit.

BitAND (valor1, valor2 [, valor n])

Parámetros

valor1	El primer número.
valor2	El segundo número.
valor n	[opcional] El enésimo número - como máximo 255 valores puede ser especificado.

Valor de Retorno

Devuelve el valor de la operación AND.

Operaciones de Bit son ejecutadas como enteros de 32-bit.

Comentarios

Recuerde que la notación hexadecimal puede ser utilizado.

BitAND devuelve 1 en cada posición de bit donde todos los argumentos son 1 en la posición correspondiente y 0 en todas las demás.

Relativo

[BitNOT](#), [BitOR](#), [BitShift](#), [BitXOR](#), [Hex](#), [BitRotate](#)

Ejemplo

$\$x = BitAND(13, 7); x == 5$ pues $1101 \text{ AND } 0111 = 0101$

$\$x = BitAND(2, 3, 6); x == 2$ pues $0010 \text{ AND } 0011 \text{ AND } 0110 = 0010$

BitNOT

Realiza una operación NOT a nivel de bit.

BitNOT (valor)

Parámetros

valor	El número para operar.
-------	------------------------

Valor de Retorno

Devuelve el valor NOT de la operación.

Operaciones con bit se realizan con enteros de 32-bit.

Comentarios

Recuerde que la notación hexadecimal puede ser utilizada.

Recuerde que en la notación de 2do complemento, BitNOT es funcionalmente equivalente a adicionar 1 y negar el resultado.

También recuerde que NOT cambia a 0 bit un 1 y viceversa.

Relativo

[BitAND](#), [BitOR](#), [BitShift](#), [BitXOR](#), [Hex](#), [BitRotate](#)

Ejemplo

$\$x = BitNot(5)$

#cs Comments:

Resultado -6 pues para números de 32-bit

5 == 000000000000000000000000000000101 binary

-6 == 1111111111111111111111111111010 binary

y el primer bit es señalado

#ce

BitOR

Realiza una operación OR a nivel de bit.

BitOR (valor1, valor2 [, valor n])

Parámetros

valor1	El primer número.
valor2	El segundo número.
valor n	[opcional] El enésimo número - como máximo 255 valores puede ser especificado.

Valor de Retorno

Devuelve los dos valores de la operación OR juntos.

Operaciones con bit se realizan con enteros de 32-bit.

Comentarios

Recuerde que la notación hexadecimal puede ser utilizada.

BitOR devuelve 0 en cada posición de bit donde todos los argumentos de entrada tienen 0 en la posición correspondiente y 1 viceversa.

Relativo

[BitAND](#), [BitNOT](#), [BitShift](#), [BitXOR](#), [Hex](#), [BitRotate](#)

Ejemplo

`$x = BitOR(3, 6) ;x == es 7 pues 0011 OR 0110 = 0111`

`$x = BitOR(3, 15, 32) ;x == 47 pues 0011 OR 1111 OR 00100000 = 00101111`

BitRotate

Realiza una operación de alternación de bit, con rotación.

BitRotate (valor , alternación [, tamaño])

Parámetros

valor	El número para ser operado.
alternación	Número de bit para rotar a la izquierda(número negativos alternan a la derecha). Si no es dado, el predefinido es 1.
tamaño	[opcional] Una cadena que determina el tamaño de la rotación, el predefinido es (16 bits). Ver más abajo.

parámetro Tamaño:

"B"	Rota bits dentro del byte de orden inferior (8 bits).
"W"	Rota bits dentro de la palabra de orden inferior (16 bits).
"D"	Rota bits dentro de la doble palabra entera (32 bits).

Valor de Retorno

Con Éxitos: Devuelve el valor rotado por el número de bits requeridos.

Al Fallar: Establece @error si el tamaño es inválido.

Operaciones de bit son realizadas con enteros de 32-bit.

Comentarios

Recuerde que la notación hexadecimal puede ser utilizada.

Relativo

[BitShift](#), [BitAND](#), [BitNOT](#), [BitOR](#), [BitXOR](#), [Hex](#)

Ejemplo

```
$x = BitRotate(7, 2)
; x == 3 pues 111b rotación doble por la derecha es 1 1100b == 28
```

```
$y = BitRotate(14, -2)
; y == 32771 pues 1110b rotación doble por la derecha de 16 bits es 1000 0000 0000 0011b ==
32771
```

```
$z = BitRotate(14, -2, "D")
; z == -2147483645 pues 1110b rotación doble por la derecha de 16 bits es 1000 0000 0000
0000 0000 0000 0000 0011b == 2147483645
```

BitShift

Realiza una operación de alternación de bit.

BitShift (valor, alternación)

Parámetros

valor	El número para ser alternado.
alternación	Número de bit par alternar a la derecha (números negativos alternan a la izquierda).

Valor de Retorno

Devuelve el valor alternado para el número requerido de bits.

Operaciones de bit son realizadas con enteros de 32 bits.

Comentarios

Recuerde que la notación hexadecimal puede ser utilizada.

La alternación a la derecha es equivalente a dividir en dos; alternación a la izquierda alterna doblando.

Relativo

[BitAND](#), [BitNOT](#), [BitOR](#), [BitXOR](#), [Hex](#), [BitRotate](#)

Ejemplo

```
$x = BitShift(14, 2)
; x == 3 pues 1110b alternación doble por la derecha es 11b == 3
```

```
$y = BitShift(14, -2)
; y == 56 pues 1110b alternación doble por la izquierda es 111000b == 56
```

```
$z = BitShift( 1, -31)
; z == -2147483648 pues es notación de segundo complemento, el
; 32do dígito desde la derecha es de signo negativo.
```

BitXOR

Realiza una operación OR exclusiva (XOR) a nivel de bit.

BitXOR (valor1, valor2 [, valor n])

Parámetros

valor1	El primer número.
valor2	El segundo número.
valor n	[opcional] El enésimo número - como máximo 255 valores puede ser especificado.

Valor de Retorno

Devuelve el valor de los parámetros XOR.

Operaciones de bit son realizadas con enteros de 32 bits.

Comentarios

Recuerde que la notación hexadecimal puede ser utilizada. BitXOR devuelve 1 en la posición del bit si existe algún que otro 1 en la posición del bit correspondiente en todos los argumentos de entrada, y 0 de otro modo.

Relativo

[BitAND](#), [BitNOT](#), [BitOR](#), [BitShift](#), [Hex](#), [BitRotate](#)

Ejemplo

`$x = BitXOR(10, 6) ;x == 12 because 1010b XOR 0110b == 1100`

`$x = BitXOR(2, 3, 6) ;x == 7 pues 0010 XOR 0011 XOR 0110 = 0111`

Cos

Calcula el coseno de un número.

Cos (expresión)

Parámetros

expresión	Valores en radianes.
-----------	----------------------

Valor de Retorno

Devuelve el coseno trigonométrico de un número.

Comentarios

$1^\circ = \pi / 180$ radianes.

Relativo

[Sin](#), [Tan](#), [ASin](#), [ACos](#), [ATan](#)

Ejemplo

```
$pi = 3.14159265358979
$x = cos($pi / 4)
$degToRad = $pi / 180
$y = Cos(90 * $degToRad) ;coseno de 90°
```

Ceiling

Devuelve un número redondeado por encima del próximo entero.

Ceiling (expresión)

Parámetros

expresión	Cualquier expresión numérica válida.
-----------	--------------------------------------

Valor de Retorno

Devuelve el número redondeado.

Comentarios

Ninguno

Relativo

[Int](#), [Number](#), [Floor](#), [Round](#)

Ejemplo

```
Dim $msg
```

```
$msg = ""  
$msg = $msg & "Ceiling(4.8) = " & Ceiling(4.8) & @CR  
$msg = $msg & "Ceiling(4.5) = " & Ceiling(4.5) & @CR  
$msg = $msg & "Ceiling(4.3) = " & Ceiling(4.3) & @CR  
$msg = $msg & "Ceiling(4) = " & Ceiling(4) & @CR  
$msg = $msg & "Ceiling(-4.3) = " & Ceiling(-4.3) & @CR  
$msg = $msg & "Ceiling(-4.5) = " & Ceiling(-4.5) & @CR  
$msg = $msg & "Ceiling(-4.8) = " & Ceiling(-4.8) & @CR  
$msg = $msg & "Ceiling(-4) = " & Ceiling(-4) & @CR
```

```
MsgBox(64, "Testeando", $msg)
```

Exp

Calcula la potencia **e** de un número.

Exp (expresión)

Parámetros

expresión	Cualquier expresión numérica válida.
-----------	--------------------------------------

Valor de Retorno

Devuelve la potencia **e** de *expresión*.

Comentarios

e es la base de los logaritmos naturales y es aproximadamente 2.71828182845905

Relativo

[Log](#)

Ejemplo

```
$x = Exp(5) ;devuelve 148.413159102577
```

Floor

Devuelve un número por debajo del entero más próximo.

Floor (expresión)

Parámetros

expresión	Cualquier expresión numérica válida.
-----------	--------------------------------------

Valor de Retorno

Devuelve el número redondeado

Comentarios

Ninguno

Relativo

[Int](#), [Number](#), [Round](#), [Ceiling](#)

Ejemplo

```
Dim $msg
```

```
$msg = ""
```

```
$msg = $msg & "Floor(4.8) = " & Floor(4.8) & @CR
$msg = $msg & "Floor(4.5) = " & Floor(4.5) & @CR
$msg = $msg & "Floor(4.3) = " & Floor(4.3) & @CR
$msg = $msg & "Floor(4) = " & Floor(4) & @CR
$msg = $msg & "Floor(-4.3) = " & Floor(-4.3) & @CR
$msg = $msg & "Floor(-4.5) = " & Floor(-4.5) & @CR
$msg = $msg & "Floor(-4.8) = " & Floor(-4.8) & @CR
$msg = $msg & "Floor(-4) = " & Floor(-4) & @CR
```

```
MsgBox(64, "Testeando", $msg)
```

Log

Calcula el logaritmo natural de un número.

Log (expresión)

Parámetros

expresión	Cualquier número positivo.
-----------	----------------------------

Valor de Retorno

Con Éxitos Devuelve el logaritmo natural del parámetro.

Al Fallar Tiende a retornar -1. #IND para parámetros no positivos.

Comentarios

@error es siempre 0

Relativo

[Exp](#)

Ejemplo

```
$x = Log(1000) ;returns 6.90775527898214  
$y = Log10(1000) ;returns 3
```

```
; Función definida de usuario para log común  
Func Log10($x)  
    Return Log($x) / Log(10) ;10 es la base  
EndFunc
```

Mod

Ejecuta una operación de módulo.

Mod (valor1, valor2)

Parámetros

valor1	El dividendo.
valor2	El divisor.

Valor de Retorno

Con Éxitos Devuelve el resto cuando valor1 es dividido por valor2.

Al Fallar Devuelve -1.#IND si el divisor es cero.

Comentarios

Esta función asume que $dividendo = (dividendo / divisor) * divisor + Mod(dividendo, divisor)$.

Esta función **no** asume que el dividendo o el divisor pueda ser representado correctamente, especialmente con números decimales.

Si un entero es pasado esta función realiza una operación integral de módulo. De otra forma, se realiza una operación de módulo con punto flotante (decimales) las cuales pueden no producir el resultado esperado.

Relativo

Ninguno.

Ejemplo

```
$n = 18  
If mod($n, 2) = 0 Then  
    MsgBox(0, "", $n & " es un número par")  
Else  
    MsgBox(0, "", $n & " es un número impar")  
EndIf
```

$\$x = \text{mod}(4, 7)$; $\$x == 4$ debido a que divisor > dividendo

$\$y = \text{mod}(1, 3/4)$; $\$y == 0.25$ debido a que divisor es un decimal

Random

Genera un número seudo-aleatorio del tipo flotante.

Random ([Mín [, Máx [, Bandera]]])

Parámetros

Mín	[opcional] El número más pequeño para ser generado. El predefinido es 0.
Máx	[opcional] El número mayor para ser generado. El predefinido es 1.
Bandera	[opcional] Si esto es establecido a 1 entonces el resultado entero será devuelto. Por defecto es un número punto flotante.

Valor de Retorno

Con Éxito: Devuelve un número seudo-aleatorio entre Mín y Máx.

Al Fallar: Devuelve 0 y la bandera de @error es establecida a 1 si los Parámetros están incorrectos.

Comentarios

Por defecto la función Random trabaja con decimales / números de punto flotante. Si desea sólo resultados con números enteros a continuación, establezca la Bandera a 1.

Si sólo un argumento es provisto, entonces se interpretará como Máx.

El resultado será en el rango de Mín a Máx INCLUSIVE cuando se utilizan enteros (sólo corto cuando Máx utiliza flotantes).

Cuando se utiliza enteros Máx-Mín debe ser inferior a 2^31 .

Comentarios sobre la base de la fuente original

Esta función utiliza la Mersenne Twister generador de números aleatorios, MT19937, escrito por Takuji Nishimura, Makoto Matsumoto, Shawn Cokus, Matthe Bellew y Isaku Wada.

El Mersenne Twister es un algoritmo para generar números aleatorios. Fue diseñado con el examen de los defectos en otros generadores. El plazo, 2^{19937 -1}, y el orden de equidistribución, 623 dimensiones, son mucho mayores. El generador también es rápido, que evita la multiplicación y la división, y se beneficia de caches y pipelines. Para obtener más información, consulte los inventores en la página web <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html> Copyright (C) 1997 - 2002, Makoto Matsumoto y Takuji Nishimura, Todos los derechos reservados.

Redistribución y uso en formato fuente y binario, con o sin modificaciones, siempre que se cumplan las siguientes condiciones:

1. Las redistribuciones del código fuente debe conservar el aviso de copyright anterior, esta lista de condiciones y la siguiente renuncia.
2. Las redistribuciones en formato binario deben reproducir el aviso de copyright anterior, esta lista de condiciones y la siguiente en la documentación y / o otros materiales suministrados con la distribución.
3. Los nombres de sus contribuyentes no podrá utilizarse para respaldar o promocionar productos derivados de este software sin el permiso previo por escrito.

Este software es proporcionado por los titulares de derechos de autor y colaboradores "tal como es" y toda las garantías expresas o implícitas, incluyendo pero no limitado a, las garantías implícitas de comerciabilidad y adecuación para un propósito en particular DETERMINADO. En ningún caso el propietario del copyright o los contribuyentes podrá ser considerada responsable de ningún daño directo, indirecto, incidental, especial, exemplar, o daños consecuentes (incluyendo, pero no limitado a, sustituir la adquisición de bienes o servicios; PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS; O interrupción de negocio), sin embargo causado y en cualquier teoría de la responsabilidad, ya sea en contrato, responsabilidad estricta, o agravio (incluyendo negligencia o de otro tipo) que surja en cualquier manera de salir del USO DE ESTE SOFTWARE, INCLUSO SI SE HA ADVERTIDO DE LA POSIBILIDAD DE TALES DAÑOS.

Relativo

[Round, SRandom](#)

Ejemplo

```
;Tapa de la moneda
If Random() < 0.5 Then ;Devuelve un valor entre 0 y 1.
    $msg = "Cara. 50% Gana"
Else
    $msg = "Cruz. 50% Pierde"
Endif
MsgBox(0,"Moneda echada", $msg)

;Rodar un dado
msgBox(0, "Rodar un dado", "Has obtenido un " & Random(1, 6, 1))
```

```

$StockPrice = 98
;En la mitad de la simulación de un juego de mercado de valores
$StockPriceChange = Random(-10, 10, 1) ; genera un entero entre -10 y 10
$StockPrice = $StockPrice + $StockPriceChange
If $StockPriceChange < 0 Then
    MsgBox(4096, "Cambio de valor", "Su valor se redujo a $" & $StockPrice)
Elseif $StockPriceChange > 0 Then
    MsgBox(4096, "Cambio de valor", "Su valor aumentó a $" & $StockPrice)
Else
    MsgBox(4096, "Cambio de valor", "Su valor quedó en $" & $StockPrice)
Endif

;Letra al azar
If Random() < 0.5 Then
    ;Mayúsculas
    $Letter = Chr(Random(Asc("A"), Asc("Z"), 1))
Else
    ;Minúsculas
    $Letter = Chr(Random(Asc("a"), Asc("z"), 1))
Endif

```

Round

Devuelve un número redondeado a un número determinado de cifras decimales.

Round (expresión [, cifras decimales])

Parámetros

expresión	Cualquier expresión numérica válida.
cifras decimales	[opcional] Número que indica cuántos lugares a la derecha del decimal se incluyen en el redondeo. Si se omite, la Round devuelve un entero.

Valor de Retorno

Devuelve el número redondeado.

Comentarios

El parámetro de *cifras decimales* puede ser negativo, que le permite redondear a unidades, decenas, centenas, etc. Tenga en cuenta que hasta quince dígitos de un número se muestran, y tome nota de que *cifras decimales* nunca se anteceden con una serie de ceros.

Relativo

[Int](#), [Number](#), [Ceiling](#), [Floor](#), [Random](#)

Ejemplo

```
$x = Round(-1.582, 1) ;devuelve -1.6  
$y = Round(3.1415, 9) ;no cambia  
$z = Round(123.5, -1) ;devuelve 120
```

Sin

Calcula el seno de un número.

Sin (expresión)

Parámetros

expresión	Valor en radianes.
-----------	--------------------

Valor de Retorno

Devuelve el seno trigonométrico de un número.

Comentarios

$1^\circ = \pi / 180$ radianes.

Relativo

[Cos](#), [Tan](#), [ASin](#), [ACos](#), [ATan](#)

Ejemplo

```
$pi = 3.14159265358979  
$x = sin($pi / 4) ; devuelve 0.707106781186547 cual es reconocido como sqrt(2)/2
```

```
$degToRad = $pi / 180  
$y = Sin(90 * $degToRad) ;sine of 90°
```

Sqrt

Calcula la raíz cuadrada de un número.

Sqrt (expresión)

Parámetros

expresión	Alguna expresión no-negativa para obtener su raíz cuadrada.
-----------	---

Valor de Retorno

Con Éxito: Devuelve la raíz cuadrada.

Al Fallar: Devuelve -1. #IND si el parámetro es negativo.

Comentarios

Para calcular una enésima raíz, use el operador de potencia: $x ^ (1/n)$

```
MsgBox(0,"La raíz cúbica de 27 es", 27 ^ (1/3) )
```

Relativo

Ninguno.

Ejemplo

```
$x = Sqrt(2) ;devuelve 1.4142135623731
$y = sqrt(9) ;devuelve 3
```

SRandom

Establece un semilla para la generación de números aleatorios.

SRandom (Semilla)

Parámetros

Semilla	Valor semilla o <i>seed</i> para generar números aleatorios. Número entre -2^31 y 2^31-1
---------	--

Valor de Retorno

Ninguno.

Comentarios

Ninguno.

Relativo

[Random](#)

Ejemplo

```
$Random(12)  
$rand=Random()
```

Tan

Calcula la tangente de un número.

Tan (expresión)

Parámetros

expresión	Valor en radianes.
-----------	--------------------

Valor de Retorno

Devuelve la trigonométrica de un número.

Comentarios

$1^\circ = \pi / 180$ radianes.

Note que Tan es "infinito" por ... $-3\pi/2, -\pi/2, \pi/2, 3\pi/2, 5\pi/2, \dots$ y tiende a devolver 1.63317787283838e+016 para esos múltiplos de la mitad de Pi.

Relativo

[Sin](#), [Cos](#), [ASin](#), [ACos](#), [ATan](#)

Ejemplo

```
$pi = 3.14159265358979
```

```
$x = tan($pi / 4)
```

```
$degToRad = $pi / 180
```

```
$y = Tan(90 * $degToRad) ;tangente de 90°
```

Referencia de Cajas de Mensajes y Diálogos

A continuación se muestra una lista completa de las funciones de Cajas de Mensajes y Diálogos disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
InputBox	Muestra una caja de entrada que requiere del usuario la entrada de una cadena.
MsgBox	Muestra un simple cuadro de mensaje con tiempo opcional para expirar.
ProgressOff	Finaliza la ventana de Progreso.
ProgressOn	Crea una ventana personalizable de barra de progreso.
ProgressSet	Establece la posición y/o texto de una ventana de Progreso creada anteriormente.
SplashImageOn	Crea una ventana con imagen personalizada.
SplashOff	Desactiva una ventana de SplashText o SplashImage.
SplashTextOn	Crea una ventana personalizada de texto.
ToolTip	Crea un tooltip en cualquier parte de la pantalla.

InputBox

Muestra una caja de entrada que requiere del usuario la entrada de una cadena.

InputBox ("título", "Prompt" [, "por defecto" [, "password char" [, ancho, [alto [, izquierda, superior [, intervalo [, hwnd]]]]]])

Parámetros

título	El título de la caja de entrada.
prompt	Un mensaje para el usuario.
por defecto	[opcional] El valor que se presentará en el input por defecto.
password char	[opcional] Un carácter que reemplazará a cualquier carácter entrado. Si usted desea que el actual carácter aparezca tipee una cadena vacía("") (por defecto) o un espacio para el primer carácter. Si usted tipea una cadena de múltiples caracteres, solamente el primer carácter es usado para enmascarar. El segundo y subsecuentes caracteres tienen una significación especial. Ver Comentarios.
ancho	[opcional] El ancho de la ventana.
altura	[opcional] El alto de la ventana.
izquierda	[opcional] El lado izquierdo de la caja de entrada. Por defecto, la caja es centrada en el escritorio.
top	[opcional] El lado superior de la caja de entrada. Por defecto, la caja es centrada en el escritorio.
intervalo	[opcional] los segundos que aguardará el InputBox para ser cancelada .
hwnd	[opcional] El apuntador de ventana para usarse como padre de este diálogo.

Valor de Retorno

Con Éxitos Devuelve la cadena que ha sido entrada.

Al Fallar Devuelve "" (cadena vacía) y fija @error a lo siguiente:

@Error 0 = La cadena revuelta es inválida.

1 = El botón Cancelar ha sido oprimido.

2 = El Intervalo de tiempo ha sido alcanzado.

3 = El InputBox falló al abrir. Esto es usualmente causado por argumentos erróneos.

4 = El InputBox no puede ser mostrado en ningún monitor.

5 = Parámetros inválidos, ancho sin alto o izquierda sin superior.

Comentarios

El InputBox es reajustable, pero hasta un tamaño mínimo de aproximadamente 190 x 115 píxeles. Por defecto el tamaño es aproximadamente 250 x 190 píxeles.

La cadena devuelta no excederá de 254 caracteres y si el input tiene retorno de carro o saltos de línea, el resultado será truncado por debajo del primer de estos caracteres encontrado.

El segundo y subsecuente caracteres del campo password pueden ser usado para restringir el input. Use un espacio para llenar el primer carácter tipeado. Colocando **M** después del primer carácter indica que el input es Mandatory; (Obligatorio) ejemplo. Usted debe entrar alguna cosa. Ningún cambio ocurre si usted presiona el botón **OK** cuando no existe ninguna cadena entrada en el InputBox. Este no cerrará y devolverá la cadena.

Usted puede también especificar la máxima longitud de caracteres a entrar en el campo de password. Solo entre un número al final del campo de password que indique la máxima longitud de la cadena.

Relativo

[MsgBox](#)

Ejemplo

```
;Colocar el input box en la esquina superior izquierda mostrando  
;los caracteres como son tipeados  
$answer = InputBox("Pregunta", "Donde usted nació?", "Planeta Tierra", "", _  
-1, -1, 0, 0)
```

```
;Preguntar al usuario por un Password. No olvidar validar el password!  
$passwd = InputBox("Chequear seguridad", "Entre su password.", "", "*")
```

```
;Pregunta al usuario para entrar 1 o 2 caracteres de respuesta. La M en el campo de password  
;indica que una cadena vacía no es aceptada y el 2 indica que la  
;respuesta no debe sobrepasar los 2 caracteres de largo.  
$value = InputBox("Testeando", "Entre un código de 1 o 2 caracteres.", "", "M2")
```

MsgBox

Muestra un simple cuadro de mensaje con tiempo opcional para expirar.

MsgBox (flag, "título", "texto" [, tiempo límite [, hwnd]])

Parámetros

flag	El flag indica el tipo de cuadro de mensaje y las posibles combinaciones de botones. Ver observaciones.
título	El título del cuadro de mensaje.
texto	El texto del cuadro de mensaje.
tiempo límite	[opcional] Tiempo de expiración en segundos. Después que el tiempo límite ha terminado el cuadro de mensaje será automáticamente cerrado. El valor predefinido es 0, lo cual no hará que expire.
hwnd	[opcional] El apuntador de la ventana para usar como el padre para esta caja de diálogo.

Valor de Retorno

Con Éxito: Devuelve el ID de el botón presionado.

Al Fallar: Devuelve -1 si el cuadro de mensaje expiró.

Botón presionado	Valor devuelto
OK	1
CANCEL	2
ABORT	3
RETRY	4
IGNORE	5
YES	6
NO	7
TRY AGAIN **	10
CONTINUE **	11

Comentarios

El parámetro de la flag puede ser una combinación de los siguientes valores:

flag decimal	Resultado Botón-relacionado	flag hexadecimal
0	botón ACEPTAR	0x0
1	ACEPTAR y Cancelar	0x1
2	Abortar, Reintentar, e Ignorar	0x2
3	Si, No, y Cancelar	0x3
4	Si y No	0x4
5	Reintentar y Cancelar	0x5
6 **	Cancelar, Intentar de nuevo, Continuar	0x6
flag decimal	Resultado de Icono-relacionado	flag hexadecimal
0	(No ícono)	0x0
16	Icono del Signo de Detener	0x10
32	Icono de Signo de Interrogación	0x20
48	Icono de signo de Exclamación	0x30
64	Icono del signo de Información que consiste en una 'i' en un círculo	0x40
flag decimal	Resultado de Botón predeterminado	flag hexadecimal
0	Primer botón es el botón por defecto	0x0
256	Segundo botón es el botón por defecto	0x100
512	Tercer botón es el botón por defecto	0x200

flag decimal	Resultado de Modalidad	flag hexadecimal
0	Aplicación	0x0
4096	Modalidad de Sistema (diálogo tiene un ícono)	0x1000
8192	Modalidad de Tarea	0x2000
flag decimal	Resultado de Misceláneos	flag hexadecimal
0	(nada especial)	0x0
262144	MsgBox tiene el atributo de SIEMPRE ENCIMA	0x40000
524288	título y texto se justifican a la derecha	0x80000

** Válido solamente en Windows 2000/XP y superiores.

Por ejemplo, para especificar un cuadro SYSTEMMODAL con los botones YES/NO los valores serán 4096+4 (o 4100) Si se usa flags hexadecimales, serían 0x1000+0x4 (o 0x1004).

Un cuadro de mensaje aparece centrado en la pantalla y se reajusta de acuerdo al texto que contiene. Sin embargo, el título podría ser truncado si el flag SYSTEMMODAL (4096) es usado.

Si "título" es igual a [Palabra clave por defecto](#), @Scriptname es usado.

Relativo

[InputBox](#), [ToolTip](#), [TrayTip](#), [SplashTextOn](#)

Ejemplo

MsgBox(4096, "Prueba", "Esta caja desaparecerá en 10 segundos", 10)

ProgressOff

Finaliza la ventana de Progreso.

ProgressOff()

Parámetros

Ninguno.

Valor de Retorno

Ninguno.

Comentarios

Ninguno.

Relativo

[ProgressSet](#), [ProgressOn](#)

Ejemplo

ProgressOff()

ProgressOn

Crea una ventana personalizable de barra de progreso.

ProgressOn ("título", "textoprincipal" [, "subtexto" [, posición x [, posición y [, opc]]]])

Parámetros

título	Título para la ventana de progreso.
textoprincipal	Texto para la etiqueta principal, en Negrita.
subtexto	[opcional] texto para Sub, Normal, Etiqueta inferior. (por defecto está en blanco)
pos x	[opcional] posición desde la izquierda (en pixeles) de ventana de progreso. (por defecto está centrado)
pos y	[opcional] posición desde arriba (en pixeles) de ventana de progreso. (por defecto está centrado)
opc	[opcional] Por defecto es 'siempre encima/con título' Sume las siguientes opciones como usted desee: 1 = sin borde, no lleva título 2 = Sin el atributo de "siempre encima" 16 = La ventana puede ser movida

Valor de Retorno

Ninguno.

Comentarios

Para omitir un parámetro opcional, dejando su valor predefinido intacto, use:

"" para los parámetros de cadena

-1 para los parámetros numéricos

Si ProgressOn es llamado mientras una ventana de progreso existe, la ventana será redibujada/movida. (Si usted desea múltiples ventanas de progresos simultáneamente, usted necesitará utilizar múltiples scripts)

Relativo

[ProgressOff](#), [ProgressSet](#)

Ejemplo

```
ProgressOn("Medidor de Progreso", "Incrementa cada segundo", "0 por ciento")
For $i = 10 to 100 step 10
    sleep(1000)
    ProgressSet($i, $i & " porciento")
Next
ProgressSet(100, "Terminado", "Completo")
sleep(500)
ProgressOff()
```

ProgressSet

Establece la posición y/o texto de una ventana de Progreso creada anteriormente.

ProgressSet (porcentaje [, "subtexto" [, "textoprincipal"]])

Parámetros

porcentaje	Porcentaje (valor entre 0. y 100.) para establecer a la barra de progreso.
subtexto	[opcional] Establece el texto para la etiqueta secundaria, Normal, Inferior.
textoprincipal	[opcional] Establece el texto para la etiqueta principal, Negrita, Superior.

Valor de Retorno

Ninguno.

Comentarios

Observe que el argumento del subtexto viene antes de texto principal.

Relativo

[ProgressOff](#), [ProgressOn](#)

Ejemplo

```
ProgressOn("Medidor de Progreso", "Incrementa cada segundo", "0 por ciento")
For $i = 10 to 100 step 10
    sleep(1000)
    ProgressSet( $i, $i & " porciento")
Next
ProgressSet(100, "Terminado", "Completo")
sleep(500)
ProgressOff()
```

SplashImageOn

Crea una ventana con imagen personalizada.

SplashImageOn ("título", "archivo" [, ancho [, alto [, posición x [, posición y [, opc]]]]])

Parámetros

título	Título para la ventana.
archivo	ruta completa\nombre de archivo de imagen (BMP, GIF, o JPG)
ancho	[opcional] ancho de ventana en pixeles (por defecto es 500)
altura	[opcional] altura en pixeles (por defecto es 400)
pos x	[opcional] posición desde la izquierda (en pixeles) de la ventana. (por defecto está centrado)
pos y	[opcional] posición desde arriba (en pixeles) de la ventana. (por defecto está centrado)
opc	[opcional] Agregar las opciones que usted quiera. Por defecto está con la opción 'siempre encima/con título' 1 = Ventana sin borde ni título 2 = Sin atributo de "siempre encima"

	16 = Ventana puede ser movida
--	-------------------------------

Valor de Retorno

Ninguno.

Comentarios

Para omitir un parámetro opcional, dejando su valor predefinido intacto, use un valor de -1.

La imagen es escalada al ancho y alto especificado y debería ser un Mapa de Bits (BMP), GIF, o imagen JPEG. Iconos (ICOs) no se mostrarán, aunque la ventana aparezca. Imágenes PNG muestran un mensaje de error al cargar, y también cuando la ruta del archivo no es válida.

Imágenes no son insertadas en el ejecutable a menos que usted utilice FileInstall.

Solo una ventana de SplashImage/Text es permitida a la vez; así que si usted desea hacer un ciclo de varias a través de múltiples imágenes/texto, simplemente llame SplashImageOn/SplashTextOn de nuevo con la nueva información.

Splash con opc=1 no puede ser movido y no puede ser activado con un clic.

Relativo

[SplashOff](#), [SplashTextOn](#)

Ejemplo

```
$destination = @Systemdir & "\ooobe\images\mslogo.jpg"  
SplashImageOn("Pantalla Splash", $destination,250,50)  
Sleep(3000)  
SplashOff()
```

SplashOff

Desactiva una ventana de SplashText o SplashImage.

SplashOff()

Parámetros

Ninguno.

Valor de Retorno

Ninguno.

Comentarios

Ninguno.

Relativo

[SplashImageOn](#), [SplashTextOn](#)

Ejemplo

```
SplashTextOn("Ejemplo", "Pantalla Splash")
Sleep(5000)
SplashOff()
```

SplashTextOn

Crea una ventana personalizada de texto.

SplashTextOn ("título", "texto" [, ancho [, altura [, posición x [, posición y [, opc [, "nombre de fuente" [, tamaño de fuente [, tamaño de espesor]]]]]]])

Parámetros

título	Título para la ventana.
texto	Texto para la ventana.
ancho	[opcional] Ancho de ventana en pixeles (por defecto 500)
altura	[opcional] Altura de ventana en pixeles (por defecto 400)
posición x	[opcional] Posición desde la izquierda (en pixeles) de la ventana. (por defecto está centrado)
posición y	[opcional] Posición desde arriba (en pixeles) de la ventana. (por defecto está centrado)
opc	[opcional] Mezcle valores como desee - predeterminadamente está 'justificado al centro/siempre encima/con título' 0 = Justificado al Centro/siempre encima/con título (por defecto) 1 = Sin borde y sin título 2 = Sin atributo de "siempre encima" 4 = Texto justificado a la izquierda 8 = Texto justificado a la derecha 16 = Ventana puede ser movida

	32 = Centra el texto verticalmente
nombre de fuente	[opcional] Fuente a utilizar. (por defecto se usa la del sistema, si fuente en este valor se deja como "" o cuando no se encuentra la fuente especificada)
tamaño de fuente	[opcional] Tamaño de fuente. (por defecto es 12; tamaños estándares son 6 8 9 10 11 12 14 16 18 20 22 24 26 28 36 48 72)
espesor de fuente	[opcional] Espesor de fuente (0 - 1000, por defecto = 400 = normal). Un valor > 1000 es tratado como cero.

Valor de Retorno

Devuelve el Apuntador de la ventana que puede ser usado en ControlSetText.

Comentarios

Para omitir un parámetro opcional, dejando su valor predefinido intacto, use:

"" para parámetros de cadena

-1 para parámetros numéricos

Solo una ventana de SplashImage/Text es permitida a la vez; así que si usted desea hacer un ciclo de varias a través de múltiples imágenes/texto, simplemente llame SplashImageOn/SplashTextOn de nuevo con la nueva información.

Incluso es mejor utilizar ControlSetText para actualizar el texto sin parpadeo ... Si el texto está centrado y con multilíneas, ControlSetText no sobrescribirá el número de líneas creadas por el SplashTextOn.

Splash con opc=1 no puede ser movido y no puede ser activado con un clic.

Nombres de fuentes estándares incluyen: Arial, Comic Sans MS, Courier New, Lucida Console, Microsoft Sans Serif, System, Tahoma, Times New Roman, y WingDings. Vea el Apéndice para obtener una lista completa.

Use @LF para mostrar varias líneas.

Relativo

[SplashOff](#), [SplashImageOn](#), [ControlSetText](#), [ToolTip](#), [MsgBox](#)

Ejemplo

```
$destination = "..\GUI\mslogo.jpg"
SplashImageOn("Splash Screen", $destination,250,50)
```

```
Sleep(3000)  
SplashOff()
```

ToolTip

Crea un tooltip en cualquier parte de la pantalla.

```
ToolTip ( "texto" [, x [, y [, "título" [, icono [, opciones]]]] ] )
```

Parámetros

texto	El texto del tooltip. (Una cadena vacía limpia o remueve un tooltip)
x	[opcional] La posición x del tooltip.
y	[opcional] La posición y del tooltip.
título	[opcional] El título para el tooltip Requiere IE5+
icono	[opcional] Icono pre-definido para mostrar, próximo al título: Requiere un título. 0 = Sin icono, 1 = Icono de Información, 2 = Icono de advertencia, 3 = Icono de Error
opciones	[opcional] Establece opciones diferentes de cómo el tooltip será mostrado (Pueden ser mezclados): 1 = Muestra el Tooltip tipo Globo Requiere IE5+ 2 = Centra el tip a las coordenadas x, y en lugar de usarlos para la esquina superior izquierda. 4 = Fuerza al tooltip que siempre esté visible confinado a las esquinas del monitor si es necesario. Si múltiples monitores son usados, entonces el tooltip aparecerá al monitor más cercano.

Valor de Retorno

Con Éxitos: Devuelve 1.

Al fallar: Devuelve 0 cuando la extensión del título es mayor 99.

Comentarios

Para omitir un parámetro opcional, dejando su valor predefinido intacto, use la palabra reservada [Default](#).

Si las coordenadas x, y son omitidas, tip será ubicado cercano a la posición del mouse. Un Tooltip aparecerá hasta que el script termine o ToolTip("") sea llamado.

Tal vez use un @CR o @LF para crear tooltips multi-líneas.

Para mostrar un ícono, usted debe especificar un título no vacío. El icono aparece en la misma fila como el título y así requiere un título para estar presente.

Si usa la bandera para centrar, el centro del tooltip estará en las coordenadas especificadas. Si usa la bandera de centrar con un Tip de Globo, el indicador será centrado en el globo y apuntará a las coordenadas especificadas.

Relativo

[TrayTip](#), [MsgBox](#), [SplashTextOn](#)

Ejemplo

*; Este creará un tooltip en la parte superior izquierda de la pantalla
ToolTip("Este es un tooltip", 0, 0)
Sleep(2000) ; Dormitar para darle tiempo al tooltip que sea mostrado*

Referencia de Funciones Varias

A continuación se muestra una lista completa de las funciones de Funciones Varias disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
AdlibRegister	Registra una función Adlib.
AdlibUnRegister	Des-registra una función adlib.
AutoItSetOption	Cambia el modo de operación de varias funciones/parámetros de AutoIt.
AutoItWinGetTitle	Devuelve el título de la ventana de AutoIt.
AutoItWinSetTitle	Cambia el título de la ventana de AutoIt.
BlockInput	Habilita/deshabilita el mouse y el teclado.
Break	Habilita o desactiva la posibilidad del usuario de finalizar el script desde el ícono de la barra de tarea.
Call	Invoca una Función de usuario contenida en una cadena como parámetro.
CDTray	Abre o cierra la bandeja del CD.
Execute	Ejecuta una expresión.
ExpandEnvStrings (Option)	Cambia cómo las cadenas literales y los símbolos % son interpretados. Por defecto las cadenas son tratadas literalmente. Esta opción le permite utilizar variables de %entorno% dentro de cadenas, ej., "El directorio temporal es: %temp%".
ExpandVarStrings (Option)	Cambia cómo las cadenas literales y los símbolos de variables/macros (\$ y @) son interpretados. Por defecto las cadenas son tratadas literalmente. Esta opción le permite utilizar variables y macros dentro de cadenas, ej., "El valor de la variable es \$var1\$".
OnAutoItExitRegister	Registra una función para ser llamada cuando AutoIt inicia.
OnAutoItExitUnRegister	Des-registra una función que es llamada cuando AutoIt finaliza.
Opt	0
SetError	Establece manualmente el valor de la macro @error.
SetExtended	Manualmente establece el valor del macro @extended.
VarGetType	Devuelve el tipo de representación interna de un dato variante.

AdlibRegister

Registra una función Adlib.

AdlibRegister ("función" [, tiempo])

Parámetros

función	El nombre de la función adlib para ser registrado.
tiempo	[opcional] Con que frecuencia, en milisegundos, se llama a la función. Por defecto es 250 ms.

Valor de Retorno

Ninguno.

Comentarios

Cada 250 ms (o *time* ms) la función especificada es llamada--típicamente para el chequeo de errores imprevistos. Por ejemplo, usted puede utilizar adlib en un script que cause un error de ventana de manera imprevisible.

La función adlib debería ser sencilla, pues se ejecutará a menudo y durante este tiempo el script estará en pausa. Además, el parámetro de tiempo debe usarse con cuidado para evitar la sobrecarga de la CPU.

Varias funciones Adlib pueden ser registradas. Al volver a registrar una función Adlib existente esta se actualizará al nuevo tiempo.

Relativo

[AdlibUnRegister](#)

Ejemplo

```
AdlibRegister("MyAdlib")
;...
Exit

Func MyAdlib()
    ;... la ejecución no debe efectuar bloqueos, evitar usar funciones ...Wait(), MsgBox(),
    InputBox()
    If WinActive("Error") Then
        ;...
    EndIf
EndFunc
```

AdlibUnRegister

Des-registra una función adlib.

AdlibUnRegister(["función"])

Parámetros

función	[opcional] El nombre de la función adlib para ser des-registrada. ver comentarios para más información.
---------	--

Valor de Retorno

Ninguno.

Comentarios

Si el nombre de la función no es especificado entonces la última función registrada será des-registrada.

Relativo

[AdlibRegister](#)

Ejemplo

```
AdlibRegister("MyAdlib")
;...
AdlibUnRegister("MyAdlib")

Func MyAdlib()
    ;... la ejecución no debe efectuar bloqueos, evitar usar funciones ...Wait(), MsgBox(),
    InputBox()
EndFunc
```

AutoItSetOption

Cambia el modo de operación de varias funciones/parámetros de AutoIt.

AutoItSetOption ("opciones" [, parámetro])

Parámetros

opciones	La opción para cambiar. Ver observaciones.
parámetro	[opcional] El valor para asignar a las opciones. El tipo y significación varía con la opción. Ver observaciones más abajo. Si el parámetro no es dado, entonces la función simplemente devuelve el valor ya asignado para esa opción. La palabra clave Default puede ser usado para resetear la opción a su valor por defecto.

Valor de Retorno

Con Éxitos: Devuelve el valor de la configuración previa para la opción.

Al Fallar: Establece @error a no-cero. Fallará si el parámetro es inválido (una opción que no existe).

Comentarios

Usted puede utilizar **Opt()** como una alternativa a **AutoItSetOption()**.

Las opciones disponibles son las siguientes:

Opción	Parámetro
CaretCoordMode	Establece la forma en que son utilizadas las coordenadas, bien sea para coordenadas absolutas o relativas a la ventana actual: 0 = coordenadas relativas a la ventana actual 1 = coordenadas absolutas a la pantalla (por defecto) 2 = coordenadas relativas a la ventana cliente de la ventana actual
ExpandEnvStrings	Cambia cómo las cadenas literales y los símbolos % son interpretados. Por defecto las cadenas son tratadas literalmente. Esta opción le permite utilizar variables de %entorno% dentro de cadenas, ej., "El directorio temporal es: %temp%". 1 = expande variables de entorno (similar a AutoIt v2) 0 = no expande variable de entorno (por defecto) Sin esta opción la manera habitual es la siguiente: "El directorio temporal es: " & EnvGet("temp")
ExpandVarStrings	Cambia cómo las cadenas literales y los símbolos de variables/macros (\$ y @) son interpretados. Por defecto las cadenas son tratadas literalmente. Esta opción le permite utilizar variables y macros dentro de cadenas, ej., "El valor de la variable es \$var1\$". 1 = expande variables (En este modo si usted desea utilizar los símbolos \$ o @ literalmente entonces debe duplicarlos: "Esto es un símbolo de dólar \$\$ ". 0 = no expande variables (por defecto)
GUICloseOnESC	Cuando ESC es presionado en una GUI el mensaje \$GUI_EVENT_CLOSE es enviado. Esta opción determina el comportamiento en on y off. 1 = Envía el mensaje \$GUI_EVENT_CLOSE cuando ESC es presionado (por defecto). 0 = No envía el mensaje \$GUI_EVENT_CLOSE cuando ESC es presionado.
GUICoordMode	Altera la posición de un control definido por GUICtrlSetPos . 1 = coordenadas absolutas (por defecto) relativas a caja de diálogos. 0 = posición relativa al inicio del control (esquina superior izquierda). 2 = Posicionamiento de la celda relativa a la celda actual. Un -1 para la izquierda o superior no incrementan el inicio. La próxima línea es -1,desplazamiento; próxima celda es con desplazamiento,-1; la celda actual es -1,-1. Obviamente "desplazamiento" no puede ser -1 que reservó para indicar el incremento. Pero si usted usa un múltiplo del

	ancho entonces puede hacer un salto o ir hacia atrás.
GUIDataSeparatorChar	Define el carácter que delimitará los subitems en GUICtrlSetData. El carácter por defecto es ' '.
GUIOnEventMode	habilita/deshabilita las notificaciones de las funciones OnEvent. 0 = (por defecto) deshabilitado. 1 = habilitado.
GUIResizeMode	Cambia el tamaño por defecto de un control. 0 = (por defecto) conserva el tamaño por defecto del control. <1024 = cualquier tamaño GUICtrlSetResizing .
GUIEventOptions	Cambia el comportamiento de eventos especiales o valores de retorno de funciones GUI. 0 = (por defecto) Comportamiento de ventanas en "on clic" "on Minimize",Restore, Maximize, Tamaño. 1 = suprime el comportamiento de ventana en "on minimize",reajuste de ventana, restore o maximize cuando damos clic en el botón. Solamente envía las notificaciones.
MouseClickDelay	Altera el tiempo de retardo entre los clic de mouse. Tiempo en milisegundos a pausar (por defecto=10).
MouseClickDownDelay	Altera el tiempo de retardo al presionar y liberar el mouse. Tiempo en milisegundos a pausar (por defecto=10).
MouseClickDragDelay	Altera el tiempo de retardo entre las operaciones de iniciar y finalizar del mouse en una operación de arrastrar. Tiempo en milisegundos a pausar (por defecto=250).
MouseCoordMode	Establece la forma en que son utilizadas las coordenadas en funciones del mouse, ya sea para coordenadas absolutas o relativas a la ventana actual: 0 = coordenadas relativas a la ventana actual 1 = coordenadas absolutas a la pantalla (por defecto) 2 = coordenadas relativas a la ventana cliente de la ventana actual
MustDeclareVars	Si esta opción es utilizada entonces todas las variables deben ser predeclaradas con Dim, Local o Global antes de poder ser usadas - evita la oportunidad de usar variables mal deletreadas causando bugs (errores). 1 = Las variables deben ser predeclaradas 0 = Las variables no necesitan ser predeclaradas (por defecto)
PixelCoordMode	establece la forma en que son utilizadas las coordenadas en funciones de píxel, ya sea para coordenadas absolutas o relativas a la ventana definida por hwnd [apuntador interno](por defecto la ventana activa): 0 = coordenadas relativas a la ventana actual 1 =coordenadas absolutas a la pantalla (por defecto) 2 = coordenadas relativas a la ventana cliente de la ventana actual
SendAttachMode	Especifica si AutoIt adjunta las entradas cuando es usada la función Send(). Cuando no están adjuntas (por defecto=0) detectando el estado de capslock/scrolllock y numlock pueda ser poco confiables en Windows NT4. Sin embargo, cuando especificas y adjunta modo=1 la sintaxis Send("{... down/up}") no trabajará y haya tal vez problemas

	enviando teclas a una ventana "colgada". ControlSend() SIEMPRE adjunta y no es afectada por este modo. 0 = no adjunta (por defecto) 1 = adjunta
SendCapslockMode	Especifica si Autolt debería almacenar el estado de capslock antes de una función Send y lo restaura posteriormente. 0 = no almacena/restaura 1 = almacena y restaura (por defecto)
SendKeyDelay	Altera la duración de brevedad entre envíos de teclas. Un valor de 0 remueve la espera completamente.
Tiempo en milisegundos a pausar (por defecto=5).	
SendKeyDownDelay	Altera la duración de tiempo entre el momento que una tecla está presionada antes de liberarla. Para aplicaciones es necesario el registrar las teclas presionadas (y en muchos juegos) tal vez necesite aumentar este valor del que viene por defecto. Un valor de 0 remueve la espera completamente.
Tiempo en milisegundos a pausar (por defecto=5).	
TCPTimeout	Define el tiempo en que las funciones de TCP declaran no comunicación. Tiempo en milisegundos antes del tiempo límite (por defecto=100).
TrayAutoPause	Script pausa cuando se da clic en el ícono de la barra de tareas. 0 = no pausa 1 = pausa (por defecto). Si no existe DefaultMenu la pausa no ocurrirá.
TrayIconDebug	Si está activado muestra la línea del script actual en la bandeja de íconos (tray ícono) en un tip, que ayuda a depurar. 0 = no información de depuración (por defecto) 1 = muestra la información
TrayIconHide	Oculta el ícono de Autolt, de la bandeja de íconos (tray ícono). Nota: El ícono inicialmente aparecerá por ~750 milisegundos. 0 = Muestra el ícono (por defecto) 1 = Oculta el ícono
TrayMenuMode	Extiende el comportamiento del ícono de la bandeja/menu. Esto puede ser una combinación(adición)de los siguientes valores. 0 = por defecto ítems de menu (Script Pause/Exit) son anexados para el menú creado por usuario; usuario revisa si los ítems son están verificados; si usted da doble clic en el ícono de la barra del sistema entonces el id de control es devuelto con el estilo-"Default" (por defecto). 1 = Menu no por defecto 2 = usuario crea ítems de verificación que no so automáticamente deschequeados si damos clic en estos 4 = No retorna el itemMenuID que tiene el estilo-"por defecto" en el menu principal si damos clic en el ícono de la barra del sistema

	8 = desactiva el auto chequeo de grupos de botones de radio
TrayOnEventMode	Habilita/deshabilita las notificaciones de las funciones de OnEvent para el icono de la barra de tareas. 0 = (por defecto) deshabilitado 1 = habilitado
WinDetectHiddenText	Especifica si el texto oculto de la ventana puede ser "visto" para funciones de coincidencia en ventana. 0 = No detecta texto oculto (por defecto) 1 = Detecta texto oculto
WinSearchChildren	Permite las rutinas de búsqueda de ventana, busque ventanas dependientes (ventana hija) como ventanas de nivel sobresaliente (top-level). 0 = Sólo busca las ventanas de nivel mayor (por defecto) 1 = Busca en todas (nivel mayor y dependientes)
WinTextMatchMode	Altera el método que es usado para coincidir con el texto de la ventana durante las operaciones de búsqueda de ventana. 1 = Completo / modo lento (por defecto) 2 = Modo rápido En modo rápido, AutoIt puede usualmente sólo "ver" en el texto de diálogo, texto en botón y títulos de algunos controles. En el modo por defecto mucho más texto puede ser visto (por instancia el contenido de la ventana de Notepad). Si tiene problemas de rendimiento cuando realiza muchas búsquedas de ventana, puede que ayude utilizar el modo rápido.
WinTitleMatchMode	Altera el método que es usado para coincidir con los títulos de las ventanas. 1 = Coincide con el título desde el inicio (por defecto) 2 = Coincide cualquier cadena en el título 3 = Coincidencia exacta de título 4 = Modo avanzado, ver Títulos de ventanas y texto (Avanzados) -1 a -4 = fuerza coincidencia en minúsculas de acuerdo con otro tipo de coincidencia.
WinWaitDelay	Altera cómo un script debería brevemente pausa después de una operación exitosa relacionada con ventanas. Tiempo en milisegundos a pausar (por defecto=250).

Relativo

Varios!

Ejemplo

; Utilize el que deseé ;Valores por defecto son listados primero

```
Opt("CaretCoordMode", 1)      ;1=absoluto, 0=relativo, 2=cliente
Opt("ExpandEnvStrings", 0)    ;0=no expandir, 1=do expandir
Opt("ExpandVarStrings", 0)    ;0=no expandir, 1=do expandir
Opt("GUICloseOnESC", 1)       ;1=ESC cierra, 0=ESC no cierra
```

```

Opt("GUICoordMode", 1)      ;1=absoluto, 0=relativo, 2=celda
Opt("GUIDataSeparatorChar","|");| es por defecto
Opt("GUIOnEventMode", 0)     ;0=deshabilitado, 1=modo OnEvent habilitado
Opt("GUIRResizeMode", 0)     ;0=no redimensionar, <1024 redimensionamiento especial
Opt("GUIEventOptions",0)    ;0=por defecto, 1=solo notificación, 2=índice tab GuiCtrlRead
Opt("MouseClickDelay", 10)   ;10 Milisegundos
Opt("MouseClickDownDelay", 10);10 Milisegundos
Opt("MouseClickDragDelay", 250);250 Milisegundos
Opt("MouseCoordMode", 1)    ;1=absoluto, 0=relativo, 2=cliente
Opt("MustDeclareVars", 0)    ;0=no, 1=predeclaración requerida
Opt("PixelCoordMode", 1)    ;1=absoluto, 0=relativo, 2=cliente
Opt("SendAttachMode", 0)     ;0=no anexa, 1=anexa
Opt("SendCapslockMode", 1)   ;1=almacena y restaura, 0=no lo hace
Opt("SendKeyDelay", 5)       ;5 milisegundos
Opt("SendKeyDownDelay", 1)   ;1 milisegundos
Opt("TCPTimeout",100)        ;100 milisegundos
Opt("TrayAutoPause",1)       ;0=no pausa, 1=Pausa
Opt("TrayIconDebug", 0)      ;0=no información, 1=línea debug de información
Opt("TrayIconHide", 0)        ;0=mostrar 1=ocultar ícono de barra de tareas
Opt("TrayMenuMode",0)        ;0=añadir, 1=menú no por defecto, 2=chequeo no automático,
4=menuItemID no retornado
Opt("TrayOnEventMode",0)      ;0=deshabilitado, 1=habilitado
Opt("WinDetectHiddenText", 0);0=no detectado, 1=detectado
Opt("WinSearchChildren", 1)   ;0=no, 1=búsqueda de hijos también
Opt("WinTextMatchMode", 1)    ;1=completo, 2=rápido
Opt("WinTitleMatchMode", 1)   ;1=iniciar, 2=subStr, 3=exacto, 4=avanzado, -1 to -4=Nocase
Opt("WinWaitDelay", 250)     ;250 milisegundos

```

AutoItWinGetTitle

Devuelve el título de la ventana de AutoIt.

AutoItWinGetTitle ()

Parámetros

Ninguno.

Valor de Retorno

Devuelve una cadena conteniendo el título de la ventana de AutoIt.

Comentarios

Ninguno.

Relativo

[AutoItWinSetTitle](#), [WinGetTitle](#)

Ejemplo

`$a = AutoItWinGetTitle()`

AutoItWinSetTitle

Cambia el título de la ventana de AutoIt.

AutoItWinSetTitle ("nuevo título")

Parámetros

nuevo título	El nuevo título de la ventana.
--------------	--------------------------------

Valor de Retorno

Ninguno.

Comentarios

La ventana de AutoIt es usualmente ocultada. El propósito para cambiar el título es para permitir que otros programas (o otros script de AutoIt) puedan interactuar con AutoIt.

Relativo

[AutoItWinGetTitle](#), [WinSetTitle](#)

Ejemplo

`AutoItWinSetTitle("Mi ventana de AutoIt")`

BlockInput

Habilita/deshabilita el mouse y el teclado.

BlockInput (flag)

Parámetros

flag	1 = Deshabilita las entradas de usuario 0 = Habilita las entradas de usuario
------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0. Si realmente esta habilitada o #requireAdmin no es usado.

Comentarios

La tabla de abajo muestra como el comportamiento de BlockInput depende de la versión de Windows; Sin embargo, presionando *Ctrl+Alt+Del* en cualquier plataforma rehabilita las entradas para las características API de Windows.

Sistema Operativo	Resultados de "BlockInput"
Windows 2000	Entrada de usuario es bloqueada y AutoIt puede simular la entrada mouse y teclado.
Windows XP	Entrada de usuario es bloqueada y AutoIt puede simular la entrada mouse y teclado. Ver los comentarios más abajo para un asunto con Windows XP SP1.
Windows Vista	Entrada de usuario es bloqueada y AutoIt puede simular la entrada mouse y teclado si #requireAdmin es usado.

Un Windows hotfix fué liberado para Windows XP SP1 que contenía un bug que prevenía la tecla Alt para que fuera enviada (via la función Send()) mientras BlockInput() esta activo. El nuevo bug de Windows es corregido en Windows XP SP2 y reciente.

BlockInput() solo afecta entradas de usuario. La entrada desde funciones como Send() o MouseMove() trabajan a pesar de ello.

Relativo

[Send](#)

Ejemplo

BlockInput(1)

```
Run("notepad")
WinWaitActive("[CLASS:Notepad]")
Send("{F5}") ;inserta tiempo y hora
```

BlockInput(0)

Break

Habilita o desactiva la posibilidad del usuario de finalizar el script desde el icono de la barra de tarea.

Break (modo)

Parámetros

modo	Estableciendo el modo de ruptura (Break): 1 = Break es habilitado (usuario puede finalizar) (por defecto) 0 = Break es deshabilitado (usuario no puede finalizar)
------	---

Valor de Retorno

Ninguno.

Comentarios

Por favor deshabilite break solamente por una buena razón.

Autolt normalmente crea un ícono en la barra de tareas de Windows cuando inicia, y dando clic derecho en el mismo el usuario puede finalizar el script. Si Break es deshabilitado (0), entonces el usuario no puede terminar el script de este modo.

Relativo

Ninguno.

Ejemplo

Break(0);deshabilita break

Call

Invoca una Función de usuario contenida en una cadena como parámetro.

Call ("función" [, parámetro1 [, parámetro2 [, parámetroN]]])

Parámetros

función	El nombre de la función de usuario a invocar.
parámetro	Argumentos que serán pasado a la función como parámetros al ser invocada.

Valor de Retorno

Con Éxitos Devuelve el valor retornado por la función indicada. @error y @extended pueden contener valores si la llamada a la función los colocó.

Al Fallar Establece @error a 0xDEAD y @extended a 0xBEEF si la función no existe o número inválido de parámetros.

Comentarios

La *función* no puede ser una función predefinida de Autolt o una función de plug-in . La función puede pasar argumentos a funciones, sin embargo, los parámetros ByRef no son soportados; pues no existe manera de recuperar un parámetro ByRef.

Un arreglo especial puede ser pasado como un único parámetro. Este arreglo debe contener en el primer elemento la cadena: "CallArgArray" y los elementos 1 - n serán pasado como argumentos separados a la función. En caso de utilizar este arreglo especial, ningún otro argumento puede ser pasado a Call(). Ver ejemplo para una demostración.

Call() en sí misma puede establecer el flag @error o la función llamada puede establecer el flag @error. Si Call() establece @error el valor será 0xDEAD y @extended será establecido también a 0xBEEF. Vea el ejemplo para una demostración de prueba para una función que no fué encontrada.

Relativo

Execute

Ejemplo

; Esto llama a una función sin aceptar argumentos.

```
Call("Test1")
```

; esto llama a una función aceptando un argumento y pasando este como argumento.

```
Call("Test2", "Mensaje desde Call()!")
```

; Esto demuestra como utilizar un arreglo especial de argumentos.

```
Global $aArgs[4]
```

\$aArgs[0] = "CallArgArray"; Esto es requerido, de otro modo, Call() no reconocerá un arreglo
conteniendo argumentos

```
$aArgs[1] = "Esto es una cadena" ; Primer parámetro es cadena
```

```
$aArgs[2] = 47 ; Segundo parámetro es un número
```

```
Global $array[2]
```

```
$array[0] = "Elemento del arreglo 0"
```

```
$array[1] = "Elemento del arreglo 1"
```

```
$aArgs[3] = $array ; Tercer parámetro es un arreglo
```

; Ya tenemos el arreglo especial, ahora invocamos la función

```
Call("Test3", $aArgs)
```

; testeá si la llamada se hace a una función que no existe. Esto muestra la forma correcta de
experimentar

; La comprobación de ambos @error y @extended contienen los valores de fallo
documentados.

```
Local Const $sFunction = "DoesNotExist"
```

```
Call($sFunction)
```

```
If @error = 0xDEAD And @extended = 0xBEEF Then MsgBox(4096, "", "Función no existe.")
```

```
Func Test1()
```

```
    MsgBox(4096, "", "Hola")
```

```
EndFunc
```

```
Func Test2($sMsg)
```

```
    MsgBox(4096, "", $sMsg)
```

```
EndFunc
```

```

Func Test3($sString, $nNumber, $aArray)
    MsgBox(4096, "", "La cedula es: " & @CRLF & $sString)
    MsgBox(4096, "", "La cadena es: "& @CRLF & $nNumber)
    For $i = 0 To UBound($aArray) - 1
        MsgBox(4096, "", "Array[" & $i & "] contenido:" & @CRLF & $aArray[$i])
    Next
EndFunc

```

CDTray

Abre o cierra la bandeja del CD.

CDTray ("drive", "status")

Parámetros

drive	La letra de la unidad de CD para controlar, en el formato D: , E: , etc.
status	Especifica el tipo de operación: abrir " open " o cerrar " closed " la bandeja.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la unidad esta bloqueada por un software quemador de CD o si la letra no corresponde a una unidad de CD.

Comentarios

CDTray funciona en unidades de CD virtuales tales como DAEMON Tools. CDTray no trabaja en unidades no-locales/mapeadas de CD pues debe estar corriendo en el computador que poseen las unidades de CD a controlar.

CDTray("X:", "close") tiende a devolver 0 aún en estilo-laptop que sólo puede cerrarse manualmente.

Relativo

[DriveGetType](#), [DriveStatus](#)

Ejemplo

; Abriendo la torre de CD en la unidad E:
CDTray("E:", "open")

Execute

Ejecuta una expresión.

Execute (cadena)

Parámetros

string	Cadena conteniendo la expresión para ser evaluada.
--------	--

Valor de Retorno

Con Éxitos Devuelve el valor de la expresión evaluada.

Al Fallar Devuelve "" (cadena vacía) con @error fijado a valor diferente de 0.

Comentarios

Ninguno.

Relativo

[Assign](#), [Eval](#), [Call](#)

Ejemplo

```
$a=1  
$v=Execute("$a+1") ; $v es establecido a 2
```

OnAutoItExitRegister

Registra una función para ser llamada cuando AutoIt inicia.

OnAutoItExitRegister("función")

Parámetros

función	El nombre de la función para llamar.
---------	--------------------------------------

Valor de Retorno

Con Éxitos: Devuelve 1. @extended puede ser establecido si es realmente registrado.

Al Fallar: Devuelve 0.

Comentarios

El código de salida puede ser recibido desde @ExitCode.

El modod de salida puede recibido desde @ExitMethod.

0	Cierre natural.
---	-----------------

1	cerrado por la función Exit.
2	cerrado por clic en 'exit' en el ícono de la barra del sistema (systray).
3	cerrado por reinicio de sesión (logoff).
4	cerrado por cierre de Windows (shutdown).

Relativo

[OnAutoItExitUnRegister](#), [Exit](#)

Ejemplo

```
OnAutoItExitRegister("MyTestFunc")
OnAutoItExitRegister("MyTestFunc2")
```

```
Sleep(1000)
```

```
Func MyTestFunc()
    MsgBox(64, "Exit Resultado 1", 'Mensaje de Exit desde MyTestFunc()')
EndFunc
```

```
Func MyTestFunc2()
    MsgBox(64, "Exit Resultado 2", 'Mensaje de Exit desde MyTestFunc()')
EndFunc
```

OnAutoItExitUnRegister

Des-registra una función que es llamada cuando AutoIt finaliza.

OnAutoItExitUnregister("función")

Parámetros

función	El nombre de la función de usuario que será llamada.
---------	--

Valor de Retorno

Con Éxitos: Devuelve 1.

Al Fallar: Devuelve 0.

Comentarios

Ninguno.

Relativo

[OnAutoItExitRegister](#)

Ejemplo

```
OnAutoItExitRegister("MyTestFunc")
OnAutoItExitRegister("MyTestFunc2")

Sleep(1000)

OnAutoItExitUnRegister("MyTestFunc")

Func MyTestFunc()
    MsgBox(64, "Exit Resultado 1", 'Mensaje de Exit desde MyTestFunc()')
EndFunc

Func MyTestFunc2()
    MsgBox(64, "Exit Resultado 2", 'Mensaje de Exit desde MyTestFunc()')
EndFunc
```

SetError

Establece manualmente el valor de la macro @error.

SetError (código [, extendido [, valor de retorno]])

Parámetros

código	El valor requerido (entero) para establecer la macro @error.
extendido	[opcional] El valor opcional valor (entero) para establecer la macro @extended. Este establece el mismo macro como la función SetExtended()
valor de retorno	[opcional] Sobrescribe el valor devuelto por defecto y devuelve este parámetro.

Valor de Retorno

Por defecto, ninguno, sin embargo si 'valor de retorno' es pasado, entonces la función devolverá ese valor.

Comentarios

Cuando entra en una función, @error es establecido a 0. A menos que SetError() sea llamado, entonces @error mantendrá 0 después que la función haya terminado. Esto significa que en el orden para establecer @error después de una función, debe ser explícitamente establecido. Este también significa que usted tal vez necesite respaldar el estado de @error en una variable si usted está utilizando un ciclo While-WEnd.

El parámetro 'extendido' es opcional. Este solamente es provisto como a manera de establecer ambos @error y @extended al mismo tiempo. Si solamente @extended se necesita establecer, entonces es recomendado usar la función SetExtended() en su lugar.

Relativo

[SetExtended](#)

Ejemplo

```
$result = myDiv(5, 0)
If @error Then
    MsgBox(4096, "Error", "Division por Cero")
Else
    MsgBox(4096, "Resultado", $result)
EndIf
Exit

Func myDiv($dividend, $divisor)
If $dividend = 0 And $divisor = 0 Then
    SetError(2) ; forma indeterminada de 0/0
ElseIf $divisor = 0 Then
    SetError(1) ; división por cero
EndIf
Return $dividend / $divisor
EndFunc
```

SetExtended

Manualmente establece el valor del macro @extended.

SetExtended (código [, valor de retorno])

Parámetros

código	El valor requerido (entero) para establecer la macro @extended.
valor de retorno	[opcional] Sobrescribe el valor de retorno predeterminado y devuelve este parámetro.

Valor de Retorno

Por defecto, ninguno, sin embargo, si 'valor de retorno' es pasado, entonces la función devolverá ese valor.

Comentarios

Al entrar en una función el macro @extended es establecido a 0. A menos que SetExtended() sea llamado, entonces @extended mantendrá 0 después de que la función haya terminado. Esto significa que en el orden para que @extended sea establecido después de una función, se debe establecer explícitamente. Esto también significa que usted necesitará hacer un respaldo del estado de @extended en una variable si lo está probando en un ciclo While-WEnd.

Relativo

[SetError](#)

Ejemplo

```
SetExtended(10)
MsgBox(4096, "Valor de @Extended es", @extended)
```

VarGetType

Devuelve el tipo de representación interna de un dato variante.

VarGetType (expresión)

Parámetros

expresión	Una expresión para revisar el tipo de representación interna.
-----------	---

Valor de Retorno

Devuelve una cadena representando el tipo de acuerdo con expresión.

Comentarios

IsInt puede devolver diferentes resultados si intenta convertir de una cadena a flotante.

Relativo

[IsInt](#), [IsFloat](#), [IsString](#), [IsBinary](#), [IsArray](#), [IsDIIStruct](#), [IsHwnd](#), [IsObj](#), [IsBool](#), [IsKeyword](#)

Ejemplo

```
$int = 1
$float = 2.0
Msgbox(0, "Tipos", "$int es " & VarGetType($int) & @CRLF & "$float es " & VarGetType($float)
)
```

Referencia de Control del Puntero

A continuación se muestra una lista completa de las funciones de Control del Puntero disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
MouseClick	Realiza una operación de clic del mouse.
MouseClickDelay (Option)	Altera el tiempo de retardo entre los clic de mouse.
MouseClickDownDelay (Option)	Altera el tiempo de retardo al presionar y liberar el mouse.
MouseClickDrag	Realiza una operación clic-y-arrastre con el mouse.
MouseClickDragDelay (Option)	Altera el tiempo de retardo entre las operaciones de iniciar y finalizar del mouse en una operación de arrastrar.
MouseCoordMode (Option)	Establece la forma en que son utilizadas las coordenadas en funciones del mouse, ya sea para coordenadas absolutas o relativas a la ventana actual:
MouseDown	Realiza un evento de presionar el mouse hacia abajo en la posición actual del mouse.
MouseGetCursor	Devuelve el número de ID del cursor actual del mouse.
MouseGetPos	Recupera la posición actual de mouse.
MouseMove	Mueve el cursor del mouse.
MouseUp	Realiza un evento de presionar el mouse hacia arriba en la posición actual del mouse.
MouseWheel	Mueve la rueda del mouse arriba o abajo. Solamente en NT/2000/XP.

MouseClick

Realiza una operación de clic del mouse.

MouseClick ("botón" [, x, y [, clics [, velocidad]]]])

Parámetros

botón	El botón a pulsar: "left" (izquierda), "right" (derecho), "middle" (intermedio), "main" (principal), "menu", "primary" (primario), "secondary" (secundario).
x, y	[opcional] Las coordenadas x/y a mover el mouse. Si las coordenadas x y no son dadas, la posición actual es usada (por defecto)

clics	[opcional] El número de veces a pulsar el mouse. Por defecto es 1.
velocidad	[opcional] La velocidad a mover el mouse en el rango 1 (más rápido) a 100 (más lento). Una velocidad de 0 moverá el ratón instantáneamente. Por defecto velocidad es 10.

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0, el botón sugerido no está en lista o se entra parámetros inválidos como x sin y.

Comentarios

Si el botón es una cadena vacía(""), el botón izquierdo será pulsado.

Si x o y es igual a la palabra reservada **Default** no ocurre movimiento en las correspondiente coordenadas.

Si el usuario ha intercambiado los botones de izquierda y derecha del mouse en el Panel de Control, entonces el comportamiento de los botones será diferente. "Left" y "derecho" siempre serán iguales, aunque hayan sido cambiados o no. El botón "primary" o "main" será el clic principal, si se han intercambiados o no los botones. Los botones "secondary" o "menu" usualmente mostrarán el menú contextual, aunque los botones hayan sido cambiados o no.

Botón	Normal	Intercambiado
""	Left	Left
"left"	Left	Left
"middle"	Middle	Middle
"right"	Right	Right
"primary"	Left	Right
"main"	Left	Right
"secondary"	Right	Left
"menu"	Right	Left

Relativo

[MouseClickDrag](#), [MouseGetPos](#), [MouseMove](#), [MouseCoordMode \(Option\)](#), [MouseClickDelay \(Option\)](#), [ControlClick](#), [MouseDown](#), [MouseUp](#), [MouseWheel](#)

Ejemplo

```
; Doble clic a la posición actual del mouse
MouseClick("left")
MouseClick("left")
```

```
; Doble clic en 0,500
```

```
MouseClick("left", 0, 500, 2)
```

; VERSION SEGURA de doble clic en 0,500 - toma en cuenta la configuración de la cuenta de usuario

```
MouseClick("primary", 0, 500, 2)
```

MouseClickDrag

Realiza una operación clic-y-arrastre con el mouse.

```
MouseClickDrag ("botón", x1, y1, x2, y2 [, velocidad])
```

Parámetros

botón	El botón a pulsar: "left" (izquierda), "right" (derecho), "middle" (intermedio), "main" (principal), "menu", "primary" (primario), "secondary" (secundario)
x1, y1	Las coordenadas x/y a iniciar la operación de arrastre.
x2, y2	Las coordenadas x/y al final de la operación de arrastre.
velocidad	[opcional] La velocidad a mover el mouse en el rango 1 (más rápido) a 100 (más lento). Una velocidad de 0 moverá el ratón instantáneamente. Por defecto la velocidad es 10.

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0, el botón no está en lista.

Comentarios

Si el botón es una cadena vacía, el botón izquierdo será pulsado.

Si el usuario ha intercambiado los botones izquierdo y derecho en el Panel de control, entonces el comportamiento de los botones será diferente. "Left" y "Right" siempre pulsa esos botones, aunque los botones sean intercambiados no. El botón "primary" o "main" será el clic principal, aunque los botones sean intercambiados. El botón "secondary" o "menu" usualmente mostrará el menú contextual, aunque los botones sean intercambiados o no.

Relativo

[MouseClick](#), [MouseGetPos](#), [MouseMove](#), [MouseCoordMode \(Option\)](#), [MouseClickDragDelay \(Option\)](#), [MouseDown](#), [MouseUp](#), [MouseWheel](#)

Ejemplo

```
; Clic derecho arrastrado desde 0,200 to 600, 700  
MouseClickDrag("left", 0, 200, 600, 700)
```

MouseDown

Realiza un evento de presionar el mouse hacia abajo en la posición actual del mouse.

MouseDown ("botón")

Parámetros

botón	El botón a pulsar: "left" (izquierda), "right" (derecho), "middle" (intermedio), "main" (principal), "menu", "primary" (primario), "secondary" (secundario)
-------	---

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0, el botón no está en lista.

Comentarios

Ver Observaciones de [MouseClick](#).

Responsabilidad de usuario: Para cada MouseDown debe eventual haber un acontecimiento correspondiente de MouseUp.

Relativo

[MouseUp](#), [MouseClick](#), [MouseClickDrag](#), [MouseGetPos](#), [MouseMove](#), [MouseCoordMode](#) (Option), [MouseClickDelay](#) (Option)

Ejemplo

```
MouseDown("left")
Sleep(100)
MouseUp("left")
```

MouseGetCursor

Devuelve el número de ID del cursor actual del mouse.

MouseGetCursor ()

Parámetros

Ninguno.

Valor de Retorno

Devuelve el número de ID del cursor:

0 = UNKNOWN (este incluye puntero e íconos de manos)
1 = APPSTARTING
2 = ARROW
3 = CROSS
4 = HELP
5 = IBEAM
6 = ICON
7 = NO
8 = SIZE
9 = SIZEALL
10 = SIZENESW
11 = SIZENS
12 = SIZENWSE
13 = SIZEWE
14 = UPARROW
15 = WAIT
16 = HAND

Comentarios

Ninguno.

Relativo

[MouseGetPos](#)

Ejemplo

sleep(2000) ;da tiempo para mover el mouse antes de reportar el ID

```
;crea un arreglo que nos dice el significado del número ID
$IDs = StringSplit("AppStarting|Arrow|Cross|Help|IBeam|Icon|No|" &_
"Size|SizeAll|SizeNESW|SizeNS|SizeNWSE|SizeWE|UpArrow|Wait|Hand", "|")
$IDs[0] = "Unknown"

$cursor = MouseGetCursor()
MsgBox(4096, "ID = " & $cursor, "Cual significa " & $IDs[$cursor])
```

MouseGetPos

Recupera la posición actual de mouse.

MouseGetPos ([dimensión])

Parámetros

Dimensión	[opcional] Un argumento opcional que determina que valor que será devuelto. Ver Valor Devuelto.
-----------	---

Valor de Retorno

Dimensión	Valores
Ninguno	Devuelve un arreglo de dos-elementos contenido las coordenadas del mouse: \$arreglo[0] = coordenada X (horizontal), \$arreglo[1] = coordenada Y (vertical)
0	Devuelve la coordenada X como un entero.
1	Devuelve la coordenada Y como un entero.

Si la dimensión no es un número, entonces @error será establecido a 1.

Comentarios

Ver MouseCoordMode para configurar posición relativa/absoluta. Si la posición es relativa, los números pueden ser negativos.

Relativo

[MouseClick](#), [MouseClickDrag](#), [MouseMove](#), [MouseCoordMode \(Option\)](#), [MouseDown](#), [MouseGetCursor](#), [MouseUp](#), [MouseWheel](#), [PixelGetColor](#)

Ejemplo

```
$pos = MouseGetPos()  
MsgBox(0, "Mouse x,y:", $pos[0] & "," & $pos[1])
```

MouseMove

Mueve el cursor del mouse.

MouseMove (x, y [,velocidad])

Parámetros

x	La coordenada x en pantalla a mover el mouse.
y	La coordenada y en pantalla a mover el mouse.
velocidad	[opcional] La velocidad a mover el mouse en el rango 1 (más rápido) a 100 (más lento). Una velocidad de 0 moverá el ratón instantáneamente. Por defecto la velocidad es 10.

Valor de Retorno

Ninguno.

Comentarios

Ninguno.

Relativo

[MouseClick](#), [MouseClickDrag](#), [MouseGetPos](#), [MouseCoordMode \(Option\)](#), [MouseDown](#), [MouseUp](#), [MouseWheel](#)

Ejemplo

```
MouseMove(10, 100)  
MouseMove(700, 700, 0)
```

MouseUp

Realiza un evento de presionar el mouse hacia arriba en la posición actual del mouse.

MouseUp ("botón")

Parámetros

botón	El botón a pulsar: "left" (izquierda), "right" (derecho), "middle" (intermedio), "main" (principal), "menu", "primary" (primario), "secondary" (secundario).
-------	--

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0, el botón no está en lista.

Comentarios

Ver Observaciones de [MouseClick](#).

Responsabilidad de usuario: Para cada MouseDown eventual debe haber un acontecimiento correspondiente de MouseUp.

Relativo

[MouseDown](#), [MouseClick](#), [MouseClickDrag](#), [MouseGetPos](#), [MouseMove](#), [MouseCoordMode \(Option\)](#)

Ejemplo

```
MouseDown("left")  
Sleep(100)  
MouseUp("left")
```

MouseWheel

Mueve la rueda del mouse arriba o abajo. Solamente en NT/2000/XP.

MouseWheel ("dirección"[, clicks])

Parámetros

dirección	"up"(arriba) o "down"(abajo)
clicks	[opcional] El número de veces a mover la rueda. Por defecto es 1.

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0, la dirección no es reconocida.

Comentarios

Esta función sólo puede funcionar en NT, 2000, XP y sistemas operativos posteriores.

Relativo

[MouseClick](#), [MouseClickDrag](#), [MouseGetPos](#), [MouseMove](#), [MouseCoordMode \(Option\)](#)

Ejemplo

; Mueve la rueda deslizante del mouse 10 veces
MouseWheel("up", 10)

Referencia de Funciones de Red

A continuación se muestra una lista completa de las funciones de Funciones de Red disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción. Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
FtpSetProxy	Establece el proxy de internet para usar en acceso FTP.
HttpSetProxy	Establece el proxy de internet para tener acceso vía http.
HttpSetUserAgent	Establece la cadena user-agent enviada con peticiones InetGet() y InetRead().
InetClose	Cierra un identificador (handle) devuelto desde InetGet().
InetGet	Descarga una archivo de Internet usando los protocolos HTTP, HTTPS o FTP
InetGetInfo	Devuelve detalles de datos para un identificador (handle) devuelto desde InetGet().
InetGetSize	Devuelve el tamaño (en bytes) de un archivo localizado en internet.
InetRead	Descarga un fichero desde internet usando los protocolos HTTP, HTTPS o FTP.
Ping	Envía pings a un host y devuelve el tiempo de ida-y-vuelta.
TCPAccept	Permite una conexión entrante intentando en un socket.
TCPCloseSocket	Cierra un socket de TCP.
TCPConnect	Crea un socket conectado un servidor existente.
TCPListen	Crea un socket en espera para una conexión entrante.
TCPNameToIP	Convierte un nombre de Internet a dirección de IP.
TCPRecv	Recibe datos de un socket conectado.
TCPSend	Envía datos en un socket conectado.
TCPShutdown	Detiene los servicios de TCP/UDP.
TCPStartup	Empieza los servicios de TCP o UDP.
TCPTimeout (Option)	Define el tiempo en que las funciones de TCP declaran no comunicación.
UDPBInd	Crea un socket delimitado para una conexión entrante.
UDPCloseSocket	Cierra un socket UDP.
UDPOpen	Abre un socket conectado para un servidor existente.

UDPRecv	Recibe datos de un socket abierto.
UDPSend	Envía un dato por un socket abierto.
UDPSshutdown	0
UDPStartup	0

FtpSetProxy

Establece el proxy de internet para usar en acceso FTP.

FtpSetProxy (modo [, "proxy:port" [, "username", ["password"]]])

Parámetros

modo	El modo proxy para usar: 0 = (por defecto) Usa la configuración actual del Internet Explorer para el Proxy. 1 = No usa proxy (acceso directo) 2 = Usa el proxy especificado
proxy:port	[opcional] La dirección y el puerto para conectarse. Ver comentarios más abajo.
username	[opcional] Si es requerido, el nombre de usuario para proxy
password	[opcional] Si es requerido, el password para proxy

Valor de Retorno

Ninguno.

Comentarios

Internet Explorer 3 o superior debe ser instalado para que esta función funcione.

Solo servidores proxy CERN son soportados.

Los servidores proxy son usualmente representados por una dirección web y un número de puerto. Muchos ISPs (proveedores de servicio) usan el número de puerto 8080. Si su proxy es "www.proxy.com" y puerto es "8080" entonces la forma de establecer la configuración es: FtpSetProxy(2, "www.proxy.com:8080")

Muchos servidores proxy no requieren un nombre de usuario y password.

Cuando usamos el modo 0 usted puede especificar un username and password y estos serán usados en el proxy configurado en Internet Explorer.

Relativo

[InetGet](#), [InetGetSize](#), [InetRead](#), [HttpSetProxy](#)

Ejemplo

```
; No usa proxy  
FtpSetProxy(1)  
  
; Usa el proxy por defecto del IE  
FtpSetProxy(0)  
  
; Usa el proxy "www-cache.myisp.net" por el puerto 8080  
FtpSetProxy(2, "www-cache.myisp.net:8080")
```

HttpSetProxy

Establece el proxy de internet para tener acceso vía http.

HttpSetProxy (modo [, "proxy:port" [, "username", ["password"]]])

Parámetros

modo	El modo de uso del proxy: 0 = (por defecto) Usa la configuración actual para el proxy en Internet Explorer. 1 = No usa proxy (acceso directo) 2 = Usar el proxy especificado
proxy:port	[opcional] La dirección y el puerto a usar. Ver observaciones más abajo.
username	[opcional] Si es requerido, el nombre de usuario para el proxy.
password	[opcional] Si es requerido, el password para el proxy.

Valor de Retorno

Ninguno.

Comentarios

Internet Explorer 3 o superior deben ser instalados para que esta función trabaje.

Solo servidores proxy CERN son soportados.

Los servidores proxy son usualmente creados con una dirección y un puerto. Muchos ISPs usan el número de puerto 8080. Si su proxy es "www.proxy.com" y su puerto es "8080" entonces usted puede establecer el proxy de esta forma:

HttpSetProxy(2, "www.proxy.com:8080")

Muchos servidores proxy no requieren nombre de usuario y password.

Cuando usamos el modo 0 usted puede especificar un username y password y estos serán usados en la configuración del proxy de Internet Explorer.

Relativo

[InetGet](#), [InetGetSize](#), [InetRead](#), [FtpSetProxy](#)

Ejemplo

```
; No usa proxy  
HttpSetProxy(1)  
  
; Usa el proxy por defecto de IE  
HttpSetProxy(0)  
  
; Usa el proxy "www-cache.myisp.net" en el puerto 8080  
HttpSetProxy(2, "www-cache.myisp.net:8080")
```

HttpSetUserAgent

Establece la cadena user-agent enviada con peticiones InetGet() y InetRead().

HttpSetUserAgent ("user agent")

Parámetros

user agent	La cadena a ser enviada como el user-agent. Por defecto el user-agent es "Autolt". Pase una cadena en blanco para resetear el user-agent anterior por defecto.
------------	--

Valor de Retorno

Ninguno.

Comentarios

Ninguno.

Relativo

[InetGet](#), [InetGetSize](#), [InetRead](#)

Ejemplo

```
HttpSetUserAgent("MiUserAgent")
```

InetClose

Cierra un identificador (handle) devuelto desde InetGet().

InetClose (identificador)

Parámetros

identificador	Un identificador devuelto por InetGet().
---------------	--

Valor de Retorno

True si el identificador fué encontrado y cerrado, False si no.

Comentarios

Los identificadores InetGet() deben ser cerrados o los recursos pueden perderse.

Al cerrar el identificador de una descarga en progreso hará que esta se cancele.

Relativo

[InetGet](#)

Ejemplo

```
Local $hDownload = InetGet("http://www.autoitscript.com/autoit3/files/beta/update.dat",
@TempDir & "\update.dat", 1, 1)
Do
    Sleep(250)
Until InetGetInfo($hDownload, 2) ; Chequea si la descarga es completada.
Local $nBytes = InetGetInfo($hDownload, 0)
InetClose($hDownload) ; Cierra el identificador para liberar el recurso.
MsgBox(0, "", "Bytes leídos: " & $nBytes)
```

InetGet

Descarga una archivo de Internet usando los protocolos HTTP, HTTPS o FTP

InetGet ("URL" [, "nombre de fichero" [, opciones [, background]]])

Parámetros

URL	La URL del archivo para descargar. Ver observaciones más abajo.
nombre de fichero	[opcional]Nombre local que tendrá el archivo descargado.
opciones	[opcional]0 = (por defecto) Obtiene el archivo de la caché local si es posible 1 = Obliga a cargar el archivo desde el sitio original. 2 = Ognora todos los errores SSL (con conexiones HTTPS). 4 = Usa ASCII con transferencias de ficheros por el protocolo FTP (Puede no estar combinado con flag 8). 8 = Usa BINARY con transferencias de ficheros por el protocolo FTP ((Puede no estar combinado con flag 4). Este es el modo de transferencia por defecto si ninguno es provisto.

	16 = Fuerza una conexión By-pass online (Ver comentarios).
background	[opcional]0 = (por defecto) Espera hasta que la descarga se complete. 1 = la función retorna inmediatamente y deja que la descarga siga en segundo plano (ver comentarios).

Valor de Retorno

- Con Éxitos El valor devuelto depende si la descarga se efectua en segundo plano:
 Segundo plano: Un identificador (handle) es devuelto. El identificador puede ser usado con InetGetInfo(). El identificador debe cerrarse con InetClose().
 En espera: El número de bytes descargados.
 Segundo plano: Un identificador (handle) es devuelto. Para determinar si ocurrió un error utilice InetGetInfo(). El identificador debe cerrarse con InetClose().
 En espera: Establece @error a no-cero y retorna 0.

Comentarios

Internet Explorer 3 o superior deben ser instalados para que esta función trabaje.

El parámetro URL debe ser la forma "http://www.somesite.com/ruta/file.html" - tal como usted lo escribe en la barra de direcciones de su navegador.

Para usar un nombre de usuario y un password simplemente colóquelos como prefijo de la URL: "username:password@", ejemplo.

"http://myuser:mypassword@www.AlgúnSitio.com"

Notas acerca del parámetro "background"

Por defecto la función espera hasta que la descarga finalice antes de retornar. si el parámetro **background** es establecido a 1 la función retorna inmediatamente y la descarga continua en segundo plano. La función InetGetInfo() puede ser usado para chequear el estado de la descarga. Esta toma un apuntador devuelto desde InetGet().

Descargas múltiples son soportadas si esta iniciado el modo segundo plano.

Para abortar una descarga utilice InetClose() y pásale un identificador devuelto desde InetGet().

Por defecto Autolt fuerza la conexión antes de comenzar la descarga. Para usuarios dial-up esta puede mostrar un prompt para online o marcar por modem (dependiendo de como el sistema es configurado). El valor opcional 16 deshabilita este comportamiento. Deshabilitar el comportamiento puede resultar útil para conexiones persistentes (Banda ancha, LAN). Sin embargo, este es también requerido para trabajar en ciertos modos en Windows Vista y Windows 7.

Relativo

[InetGetSize](#), [InetRead](#), [InetGetInfo](#), [InetClose](#), [HttpSetProxy](#), [FtpSetProxy](#), [HttpSetUserAgent](#)

Ejemplo

```
InetGet("http://www.mozilla.org", @TempDir & "\mozilla.html")
InetGet("http://www.autoitscript.com", @TempDir & "autoitscript.html", 1)
InetGet("ftp://ftp.mozilla.org/pub.mozilla.org/README", @TempDir & "\Mozilla-README.txt",
1)

; Ejemplo avanzado - descargando en segundo plano
Local $hDownload = InetGet("http://www.autoitscript.com/autoit3/files/beta/update.dat",
@TempDir & "\update.dat", 1, 1)
Do
    Sleep(250)
Until InetGetInfo($hDownload, 2) ; Chequea si la descarga es completada
Local $nBytes = InetGetInfo($hDownload, 0)
InetClose($hDownload) ; Cierra el identificador para liberar el recurso
MsgBox(0, "", "Bytes leídos: " & $nBytes)
```

InetGetInfo

Devuelve detalles de datos para un identificador (handle) devuelto desde InetGet().

InetGetInfo([identificador [, índice]])

Parámetros

identificador	[opcional] Un identificador devuelto desde InetGet().
índice	[opcional] El índice del dato a recibir. Si este valor es -1 un arreglo recibiendo todos los datos mostrados abajo será mostrado. 0 - Bytes leídos hasta el momento (esto es actualizado mientras la descarga progres). 1 - El tamaño de la descarga (esto no siempre está presente). 2 - Establecido a True si la descarga se completa, False si la descarga está en progreso. 3 - Establecido a True si la descarga fue satisfactoria. Si es False entonces el próximo miembro de dato será establecido a no-cero. 4 - El valor error para la descarga. El valor en sí mismo es arbitrario. Comprobando que este valor sea no-cero es suficiente para determinar si un error ocurrió. 5 - El valor extendido para la descarga. El valor es arbitrario y es primariamente solo útil para desarrolladores de AutoIt.

Valor de Retorno

Con Éxitos - El dato requerido.

Al Fallar - Una cadena en blanco "" y @error es establecido a no-cero.

Comentarios

Si es llamado sin argumentos entonces es devuelto es número total de descargas activas.

Esta función puede ser llamada dentro de un ciclo para consultar el número de bytes descargados o para pausar mientras la descarga es completada.

Relativo

[InetGet](#)

Ejemplo

```
#Alternate
Local $hDownload = InetGet("http://www.autoitscript.com/autoit3/files/beta/update.dat",
@TempDir & "\update.dat", 1, 1)
Do
    Sleep(250)
Until InetGetInfo($hDownload, 2) ; Chequea si la descarga es completada
Local $aData = InetGetInfo($hDownload) ; Obtiene toda la información
InetClose($hDownload) ; Cierra el identificador para liberar el recurso
MsgBox(0, "", "Bytes leídos: " & $aData[0] & @CRLF &_
    "Tamaño: " & $aData[1] & @CRLF &_
    "¿Completado?: " & $aData[2] & @CRLF &_
    "¿Satisfactorio?: " & $aData[3] & @CRLF &_
    "@error: " & $aData[4] & @CRLF &_
    "@extended: " & $aData[5] & @CRLF)
```

InetGetSize

Devuelve el tamaño (en bytes) de un archivo localizado en internet.

InetGetSize ("URL" [, opciones])

Parámetros

URL	URL del archivo a descargar. Ver observaciones más abajo.
opciones	[opcional] 0 = (por defecto) Obtiene el archivo de la caché local si es posible 1 = Obliga a cargar el archivo desde el sitio original. 2 = Ignora todos los errores SSL (con conexiones HTTPS). 4 = Usa ASCII con transferencias de ficheros por el protocolo FTP (Puede no estar combinado con flag 8). 8 = Usa BINARY con transferencias de ficheros por el protocolo FTP ((Puede no estar combinado con flag 4). Este es el modo de transferencia por defecto si ninguno es provisto.

Valor de Retorno

Con Éxitos Devuelve el tamaño de el archivo en bytes.

Al Fallar Devuelve 0 y establece el valor de @error a no-cero.

Comentarios

Internet Explorer 3 o superior deben ser instalados para que esta función trabaje. (Para ftp:// URLs IE 5 es requerido!)

El parámetro URL debe ser la forma "http://www.somesite.com/ruta/file.html" - tal como usted lo escribe en la barra de direcciones de su navegador.

Para usar un nombre de usuario y un password simplemente colóquelos como prefijo de la URL: "username:password@", ejemplo.

"http://myuser:mypassword@www.AlgúnSitio.com"

No todos los servidores pueden tomar correctamente el tamaño del archivo, especialmente cuando es usado un servidor proxy.

Relativo

[InetGet](#), [InetRead](#), [HttpSetProxy](#), [FtpSetProxy](#), [HttpSetUserAgent](#)

Ejemplo

```
Local $nsize = InetGetSize("http://www.autoitscript.com/autoit3/files/beta/update.dat")
MsgBox(0, "Tamaño del fichero remoto:", $nsize)
```

InetRead

Descarga un fichero desde internet usando los protocolos HTTP, HTTPS o FTP.

InetRead ("URL" [, opciones])

Parámetros

URL	La dirección URL del fichero a descargar. Ver Comentarios más abajo.
opciones	[opcional] 0 = (por defecto) Obtiene el fichero desde la caché local si es posible. 1 = Fuerza a descargar desde un sitio remoto. 2 = Ignora todos los errores SSL (con conexiones HTTPS). 4 = Usa modo ASCII cuando transfiere ficheros con el protocolo FTP (Puede no estar combinado con flag 8). 8 = Usa modo BINARY cuando transfiere ficheros con el protocolo FTP (Puede no estar combinado con flag 4). Este es el modo de transferencia por defecto si ninguno es provisto. 16 = Obligar By-pass para la conexión online (Ver comentarios).

Valor de Retorno

Con Éxitos: Una cadena binaria @extended establecido al número de bytes descargados.

Al Fallar: Establece @error a no-cero y devuelve una cadena en blanco.

Comentarios

Internet Explorer 3 o superior deben ser instalados para que esta función trabaje.

El parámetro URL debe ser en la forma "http://www.somesite.com/path/file.html" - justamente como usted lo pondría en la barra de direcciones de su navegador.

Para utilizar un username y password cuando se conecte simplemente colóquelo como prefijo de la url así "username:password@", e.j.

"http://myuser:mypassword@www.somesite.com"

El dato returnedo esta en formato binario. La función [BinaryToString\(\)](#) pede ser usada para convertir el dato a cadena.

Por defecto Autolt fuerza la conexión antes de comenzar la descarga. Para usuarios dial-up esta puede mostrar un prompt para online o marcar por modem (dependiendo de como el sistema es configurado). El valor opcional 16 deshabilita este comportamiento. Deshabilitar el comportamiento puede resultar útil para conexiones persistentes (Banda ancha, LAN). Sin embargo, este es también requerido para trabajar en ciertos modos en Windows Vista y Windows 7.

Relativo

[InetGet](#), [InetGetSize](#), [HttpSetProxy](#), [FtpSetProxy](#), [HttpSetUserAgent](#)

Ejemplo

```
Local $sData = InetRead("http://www.autoitscript.com/autoit3/files/beta/update.dat")
Local $nBytesRead = @extended
MsgBox(4096, "", "Bytes leídos: " & $nBytesRead & @CRLF & @CRLF & BinaryToString($sData))
```

Ping

Envía pings a un host y devuelve el tiempo de ida-y-vuelta.

Ping (dirección/nombre del host [, tiempoExpiración])

Parámetros

dirección/nombre del host	Puede ser por ej. "www.autoitscript.com" o "87.106.244.38"
tiempoExpiración	[opcional] Es el tiempo a esperar para una respuesta en milisegundos (por defecto es 4000).

Valor de Retorno

- Con Éxito: Devuelve el el tiempo de ida-y-vuelta en milisegundos (mayor que 0).
Al Fallar: Devuelve 0 si el host no admite ping u otros errores de red ocurridos y establece @error. (Ver Comentarios)

Comentarios

Cuando la función falla (devuelve 0) @error contiene información extendida:

- 1 = Host está fuera de línea (offline)
- 2 = Host es inalcanzable
- 3 = Mal destino
- 4 = Otros errores

Relativo

Ninguno.

Ejemplo

```
$var = Ping("www.AutoItScript.com",250)
If $var Then; también posible: If @error = 0 Then ...
    MsgBox(0,"Status","Online, ida y vuelta fué de:" & $var)
Else
    MsgBox(0,"Status","Un error ocurrido con el número: " & @error)
Endif
```

TCPAccept

Permite una conexión entrante intentando en un socket.

TCPAccept (SocketPrincipal)

Parámetros

SocketPrincipal	El principal identificador del socket (ID del Socket) como es devuelta por la función TCPListen .
-----------------	---

Valor de Retorno

- Con Éxito: Devuelve el identificador conectado del socket.
Al Fallar: Devuelve -1 y establece @error a:
@error: valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[TCPStartup](#), [TCPListen](#), [TCPTimeout \(Option\)](#), [TCPCloseSocket](#), [TCPRecv](#)

Ejemplo

```
;SERVIDOR!! Iniciame primero !!!!!!!!
$g_IP = "127.0.0.1"

; Iniciar los Servicios de TCP
=====
TCPStartUp()

; Crea un "SOCKET" escucha
=====
$MainSocket = TCPListen($g_IP, 65432, 100 )
If $MainSocket = -1 Then Exit

; buscar una conexión cliente
;-----
While 1
$ConnectedSocket = TCPAccept( $MainSocket)
If $ConnectedSocket >= 0 Then
  msgbox(0,"","mi servidor - Cliente Conectado")
  exit
EndIf
Wend
```

TCPCloseSocket

Cierra un socket de TCP.

TCPCloseSocket (socket)

Parámetros

socket	El identificador de socket (ID del Socket) como es devuelto por las funciones TCPListen o TCPAccept .
--------	---

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve -1 y establece @error a:

@error: valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[TCPStartup](#), [TCPListen](#), [TCPAccept](#), [TCPShutdown](#)

Ejemplo

```
#include <GUIConstantsEx.au3>
#include <WindowsConstants.au3>
#include <ButtonConstants.au3>

Opt('MustDeclareVars', 1)

;=====
;=====
;SERVIDOR!! Iniciame primero !!!!!!!!
;=====
;=====

; Inicializa una variable para representar una conexión
;=====
Global $ConnectedSocket = -1

Global $MainSocket

Ejemplo()

Func Ejemplo()
    OnAutoItExitRegister("Cleanup")
    Local $g_IP, $RogueSocket, $GOOEY, $edit, $input, $butt, $msg
    Local $ret, $recv

    $g_IP = "127.0.0.1"

    ; Inicia los servicios TCP
    ;=====
    TCPStartup()

    ; Crea un "SOCKET" escuchando.
    ;=====
    $MainSocket = TCPListen($g_IP, 65432, 100)
    If $MainSocket = -1 Then Exit
    $RogueSocket = -1

    ; Crea un GUI para chatear
    ;=====
    $GOOEY = GUICreate("mi servidor", 300, 200)
    $edit = GUICtrlCreateEdit("", 10, 40, 280, 150, $WS_DISABLED)
    $input = GUICtrlCreateInput("", 10, 10, 200, 20)
    $butt = GUICtrlCreateButton("Enviar", 210, 10, 80, 20, $BS_DEFPUSHBUTTON)
    GUISetState()
```

```

; Ciclo de mensajes del GUI
;=====
While 1
    $msg = GUIGetMsg()

    ; GUI Cerrado
    ;-----
    If $msg = $GUI_EVENT_CLOSE Then ExitLoop

    ; al presionar ENVIAR
    ;-----
    If $msg = $butt Then
        If $ConnectedSocket > -1 Then
            $ret = TCPSend($ConnectedSocket, GUICtrlRead($input))
            If @error Or $ret < 0 Then
                ; ERROR OCURRIDO, CERRAR SOCKET Y REINICIARLO ConnectedSocket to -1
                ;-----
                TCPCloseSocket($ConnectedSocket)
                WinSetTitle($GOOEY, "", "mi servidor - Cliente Desconectado")
                $ConnectedSocket = -1
            Elseif $ret > 0 Then
                ; ACTUALIZA EL CONTROL EDIT CON LOS DATOS QUE ENVIAMOS
                ;-----
                GUICtrlSetData($edit, GUICtrlRead($edit) & GUICtrlRead($input) & @CRLF)
            EndIf
        EndIf
        GUICtrlSetData($input, "")
    EndIf

    If $RogueSocket > 0 Then
        $recv = TCPRecv($RogueSocket, 512)
        If Not @error Then
            TCPCloseSocket($RogueSocket)
            $RogueSocket = -1
        EndIf
    EndIf

    ; Si no hay conexión buscar una
    ;-----
    If $ConnectedSocket = -1 Then
        $ConnectedSocket = TCPAccept($MainSocket)
        If $ConnectedSocket < 0 Then
            $ConnectedSocket = -1
        Else
            WinSetTitle($GOOEY, "", "mi servidor - Cliente Conectado")
        EndIf

        ; Si está conectado leer datos
        ;-----
        Else
            ; EJECUTAR UN ACEPTAR INCONDICIONAL EN CASO QUE OTRO CLIENTE INTENTE
            CONECTAR

```

```

;-----
$RogueSocket = TCPAccept($MainSocket)
If $RogueSocket > 0 Then
    TCPSend($RogueSocket, "~~rechazado")
EndIf

$recv = TCPRecv($ConnectedSocket, 512)

If $recv <> "" And $recv <> "~~adios" Then
    ; ACTUALIZAR EL CONTROL DE EDIT CON LOS DATOS RECIBIDOS
;-----
GUICtrlSetData($edit, GUICtrlRead($edit) & ">" & $recv & @CRLF)

Elseif @error Or $recv = "~~adios" Then
    ; ERROR OCCURRED, CLOSE SOCKET AND RESET ConnectedSocket to -1
;-----
WinSetTitle($GOOEY, "", "mi servidor - Cliente Desconectado")
TCPCloseSocket($ConnectedSocket)
$ConnectedSocket = -1
EndIf
EndIf
WEnd

GUIDelete($GOOEY)
EndFunc ;==>Ejemplo

Func Cleanup()
;AL SALIR EL SCRIPT cerrar todos los sockets abiertos y apagar el servicio de TCP
;-----
If $ConnectedSocket > -1 Then
    TCPSend($ConnectedSocket, "~~adios")
    Sleep(2000)
    TCPRecv($ConnectedSocket, 512)
    TCPCloseSocket($ConnectedSocket)
EndIf
TCPCloseSocket($MainSocket)
TCPShutdown()
EndFunc ;==>Cleanup

```

TCPConnect

Crea un socket conectado un servidor existente.

TCPConnect (*IPAddr*, *puerto*)

Parámetros

IPAddr	Dirección del Protocolo de Internet (IpV4) como "192.162.1.1"
puerto	puerto en el que el socket creado será conectado.

Valor de Retorno

- Con Éxito: Devuelve identificador principal de socket.
- Al Fallar: Devuelve -1 o 0 y establece @error a:
- @error: 1 IPAddr es incorrecto
 - 2 puerto es incorrecto.
- valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Esta función es usada por un cliente para comunicar con el servidor.

Relativo

[TCPStartup](#), [TCPListen](#), [TCPTimeout \(Option\)](#), [TCPRecv](#), [TCPSend](#)

Ejemplo

```
;CLIENTE!!!!!! Inicie el SERVIDOR Primero... dummy!!
$g_IP = "127.0.0.1"

; Iniciar los servicios TCP
=====
TCPStartUp()

; Conectar a un "SOCKET" escucha
=====
$socket = TCPConnect( $g_IP, 65432 )
If $socket = -1 Then Exit
```

TCPListen

Crea un socket en espera para una conexión entrante.

TCPListen (IPAddr, puerto [, MaxPendingConnection])

Parámetros

IPAddr	Dirección de Protocolo de Internet(IpV4) como "192.162.1.1"
puerto	puerto en el que el socket será conectado.
MaxPendingConnection	[opcional] Longitud máxima de espera para conexiones pendientes. Por defecto el valor máximo razonables será establecido.

Valor de Retorno

- Con Éxito: Devuelve el identificador del socket principal.

- Al Fallar: Devuelve -1 o 0 y establece @error a:
 @error: 1 IPAddr es incorrecto
 2 puerto es incorrecto.
 valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[TCPStartup](#), [TCPConnect](#), [TCPAccept](#), [TCPTIMEOUT \(Option\)](#), [TCPCloseSocket](#), [TCPSEND](#), [TCPShutdown](#)

Ejemplo

```
;SERVIDOR!! Iniciar me primero !!!!!!!!
$g_IP = "127.0.0.1"

; Iniciar los servicios TCP
=====
TCPStartUp()

; Conectar a un "SOCKET" escucha
=====
$MainSocket = TCPListen($g_IP, 65432, 100 )
If $MainSocket = -1 Then Exit
```

TCPNameToIP

Convierte un nombre de Internet a dirección de IP.

TCPNameToIP (nombre)

Parámetros

nombre	Cadena conteniendo un nombre de Internet.
--------	---

Valor de Retorno

- Con Éxito: Devuelve una cadena conteniendo la dirección de IP correspondiente al nombre.
 Al Fallar: Devuelve "" y establece @error:
 @error: valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

TCPStartup es necesario antes de llamar esta función.

Relativo

[TCPStartup](#)

Ejemplo

TCPStartup()

```
msgbox(0,"IP???",TCPNameToIP("LaPCCercaDeUsted"))
```

TCPRcv

Recibe datos de un socket conectado.

TCPRcv (socket_Principal, long_máx [, bandera])

Parámetros

socket_Principal	El identificador del socket conectado (ID del Socket) como es devuelto por las funciones TCPAccept o TCPConnect .
long_máx	Número máx # de caracteres a recibir.
bandera	[opcional] Fuerza la función a devolver datos binarios si se establece a 1 (por defecto está en 0, y auto-detectará entre binario/cadena).

Valor de Retorno

Con Éxito: Devuelve binario/cadena enviado por el socket conectado.

Al Fallar: Devuelve "" y establece @error a:

@error: -1 error de Socket

valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Por razones de compatibilidad esta función intentará devolver cadenas por defecto. Si carácter nulos (0x00) son recibidos entonces el valor devuelto será un tipo binario.

Para forzar a la función que siempre devuelva datos binarios (la opción más sensible) entonces use el parámetro "bandera" en 1.

Relativo

[BinaryLen](#), [BinaryMid](#), [TCPStartup](#), [TCPAccept](#), [TCPConnect](#), [TCPTimeout](#) (Option)

Ejemplo

```

#include <GUIConstantsEx.au3>

Opt('MustDeclareVars', 1)

;=====
;=====
;SERVIDOR!! Iniciame primero !!!!!!!!
;=====
;=====

Example()

Func Example()
    ; Establecer información reusable
    ; Establecer su IP pública (@IPAddress1) aquí.
    ; Local $szServerPC = @ComputerName
    ; Local $szIPADDRESS = TCPNameToIP($szServerPC)
    Local $szIPADDRESS = @IPAddress1
    Local $nPORT = 33891
    Local $MainSocket, $GOOEY, $edit, $ConnectedSocket, $szIP_Accepted
    Local $msg, $recv

    ; Inicia los servicios TCP
    ;=====
    TCPStartup()

    ; Crea un "SOCKET" escuchando.
    ; Usando su dirección IP y puerto 33891.
    ;=====
    $MainSocket = TCPListen($szIPADDRESS, $nPORT)

    ; Si la creación del Socket falla, terminará.
    If $MainSocket = -1 Then Exit

    ; Crea un GUI para mensajes
    ;=====
    $GOOEY = GUICreate("Mi Servidor (IP: " & $szIPADDRESS & ")", 300, 200)
    $edit = GUICtrlCreateEdit("", 10, 10, 280, 180)
    GUISetState()

    ; Inicializa una variable para representar una conexión
    ;=====
    $ConnectedSocket = -1

    ; Espera y acepta un conexión
    ;=====
    Do
        $ConnectedSocket = TCPAccept($MainSocket)
    Until $ConnectedSocket <> -1

```

```

; Obtiene un IP del cliente que se conecta
$szIP_Accepted = SocketToIP($ConnectedSocket)

; Ciclo de mensajes del GUI
;=====
While 1
    $msg = GUIGetMsg()

    ; GUI Cerrado
    ;-----
    If $msg = $GUI_EVENT_CLOSE Then ExitLoop

    ; Intentar recibir (hasta) 2048 bytes
    ;-----
    $recv = TCPRecv($ConnectedSocket, 2048)

    ; Si recibir falla con @error entonces el socket se desconectará
    ;-----
    If @error Then ExitLoop

    ; Actualiza el control de edit con lo que recibimos
    ;-----
    If $recv <> "" Then GUICtrlSetData($edit,
        $szIP_Accepted & " > " & $recv & @CRLF & GUICtrlRead($edit))
WEnd

If $ConnectedSocket <> -1 Then TCPCloseSocket($ConnectedSocket)

TCPShutdown()
EndFunc ;==>Ejemplo

; Función para devolver una dirección IP de un socket conectado.
;-----
Func SocketToIP($SHOCKET)
    Local $sockaddr, $aRet

    $sockaddr = DllStructCreate("short;ushort;uint;char[8]")

    $aRet = DllCall("Ws2_32.dll", "int", "getpeername", "int", $SHOCKET, _
        "ptr", DllStructGetPtr($sockaddr), "int*", DllStructGetSize($sockaddr))
    If Not @error And $aRet[0] = 0 Then
        $aRet = DllCall("Ws2_32.dll", "str", "inet_ntoa", "int", DllStructGetData($sockaddr, 3))
        If Not @error Then $aRet = $aRet[0]
    Else
        $aRet = 0
    EndIf

    $sockaddr = 0

```

```
Return $aRet  
EndFunc ;==>SocketToIP
```

TCPSend

Envía datos en un socket conectado.

TCPSend (socketPrincipal, datos)

Parámetros

socketPrincipal	El identificador del socket conectado (ID del Socket) como es devuelto por la función TCPConnect .
datos	binario/cadena para ser enviado al socket conectado.

Valor de Retorno

Con Éxito: Devuelve el número de bytes enviados al socket conectado.
Al Fallar: Devuelve "" y establece @error a:
@error: -1 error de Socket
valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[TCPStartup](#), [TCPConnect](#), [TCPListen](#), [TCPTIMEOUT \(Option\)](#)

Ejemplo

```
#include <GUIConstantsEx.au3>  
Opt('MustDeclareVars', 1)  
  
=====  
=====  
;CLIENTE! Iniciar me después del Servidor!!!!!!!  
=====  
=====
```

Example()

```
Func Example()  
; Alguna información reusable  
;  
Local $ConnectedSocket, $szData
```

```

; Establecer $szIPADDRESS a cual sea el servidor. Cambiaremos un nombre de PC en una
dirección IP
; Local $szServerPC = @ComputerName
; Local $szIPADDRESS = TCPNameToIP($szServerPC)
Local $szIPADDRESS = @IPAddress1
Local $nPORT = 33891

; Inicia los Servicios de TCP
;=====
TCPStartup()

; Inicializa una variable para representar una conexión
;=====
$ConnectedSocket = -1

; Intenta conectar al SERVIDOR a su IP y PUERTO 33891
;=====
$ConnectedSocket = TCPConnect($szIPADDRESS, $nPORT)

; Si hay un error... mostrarlo
If @error Then
    MsgBox(4112, "Error", "TCPConnect falló con error WSA: " & @error)
    ; Si no hay error de ciclo una entrada para datos
    ; se enviará al SERVIDOR
Else
    ;Loop forever asking for data to send to the SERVER
    While 1
        ; InputBox para datos a transmitir
        $szData = InputBox("Datos para el Servidor", @LF & @LF & "Introduzca los datos a
transmitir al SERVIDOR:")
        ; Si se cancela el InputBox o se deja en blanco saldremos del ciclo
        If @error Or $szData = "" Then ExitLoop

        ; Deberíamos tener datos en $szData... intentemos enviarlo a través de nuestro socket
        ; conectado.
        TCPSend($ConnectedSocket, $szData)

        ; Si el envío falló con @error entonces el socket será desconectado
        ;=====
        If @error Then ExitLoop
    WEnd
EndIf
EndFunc ;==>Ejemplo

```

TCPShutdown, UDPShutdown

Detiene los servicios de TCP/UDP.

TCPShutdown ()
UDPSHUTDOWN ()

Parámetros

Ninguno.

Valor de Retorno

Con Éxito: Devuelve 1.
Al Fallar: Devuelve 0 y establece @error a:
@error: valor de retorno de windows API WSACleanup (ver [MSDN](#)).

Comentarios

Un script debe llamar una función TCPShutdown() para cada llamada exitosa de TCPStartup(). UDPSHUTDOWN() es simplemente un alias de TCPShutdown().

Relativo

[TCPStartup](#), [TCPListen](#), [TCPCloseSocket](#), [UDPCloseSocket](#)

Ejemplo

TCPStartup() ; Inicia los servicios TCP

TCPShutdown () ; Detiene los servicios TCP

TCPStartup, UDPStartup

Empieza los servicios de TCP o UDP.

TCPStartup ()
UDPStartup ()

Parámetros

Ninguno.

Valor de Retorno

Con Éxito: Devuelve 1.
Al Fallar: Devuelve 0 y establece @error a:
@error: valor de retorno de windows API WSAStartup (ver [MSDN](#)).

Comentarios

Aquí debe haber una llamada de TCPShutdown() para evitar consumo de memoria. Un script debe llamar a TCPShutdown() para cada llamada exitosa de TCPStartup(). UDPStartup() es simplemente un alias de TCPStartup().

Relativo

[TCPShutdown](#), [TCPListen](#), [TCPCloseSocket](#), [UDPCloseSocket](#), [TCPAccept](#), [TCPConnect](#), [TCPNameToIP](#), [TCPRecv](#), [TCPSend](#)

Ejemplo

TCPStartup () ; Iniciar los servicios TCP

UDPBIND

Crea un socket delimitado para una conexión entrante.

UDPBIND (*IPAddr*, *puerto*)

Parámetros

IPAddr	Protocolo de internet de dirección punteada(IpV4) como "192.162.1.1"
puerto	puerto al cual el socket creado será confinado.

Valor de Retorno

- Con Éxitos Devuelve un arreglo : \$arreglo[1] conteniendo el socket real, \$arreglo[2] conteniendo la dirección IP y \$arreglo[3] conteniendo el puerto. Necesitaremos esta información en llamadas subsecuentes a UDPSend(), donde será pasado esta estructura/arreglo del socket.
- Al Fallar Establece @error a:
- @error: 1 IPAddr es incorrecto.
 2 port es incorrecto
 valor de retorno de windows API WSAGetError(ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[UDPRecv](#), [UDPOpen](#), [UDPCloseSocket](#), [UDPSend](#)

Ejemplo

*;SERVIDOR!! Iniciarme primero!!!!!!!!!!!!!!
\$g_IP = "127.0.0.1"*

```

; Iniciando los servicios UDP
=====
UDPStartUp()

; Crea un Listening "SOCKET"
=====
$socket = UDPBind($g_IP, 65432)
If @error <> 0 Then Exit

```

UDPCloseSocket

Cierra un socket UDP.

UDPCloseSocket (socketarray)

Parámetros

socketarray	El socket/arreglo devuelto por las funciones UDPBInd o UDPOpen .
-------------	--

Valor de Retorno

Con Éxitos Devuelve 1.
 Al Fallar Devuelve 0 y establece @error a:
 @error: -1, -2 or -3 socketarray inválido.
 valor de retorno de windows API WSAGetError(ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[UDPBInd](#), [UDPOpen](#), [TCPShutdown](#), [TCPStartup](#)

Ejemplo

```

;SERVIDOR!! Iniciar me primero!!!!!!!!!!!!!!
$g_IP = "127.0.0.1"

```

```

; Iniciando los servicios UDP
=====
UDPStartUp()

; Registra una función de limpieza
OnAutoItExitRegister("Cleanup")

; Crea un Listening "SOCKET"
=====
$socket = UDPBind($g_IP, 65432)

```

If @error <> 0 Then Exit

;--- Su código aquí

```
Func Cleanup()
    UDPCloseSocket($socket)
    UDPShutdown()
EndFunc
```

UDPOpen

Abre un socket conectado para un servidor existente.

UDPOpen (IPAddr, puerto [, flag])

Parámetros

IPAddr	Protocolo de internet de dirección punteada(IPV4) como "192.162.1.1"
puerto	puerto al cual el socket creado será conectado.
flag	[opcional] 0 (Por defecto) - La opciones adicionales no son establecidas. 1 - Permitir broadcasting (difusión) en la dirección "255.255.255.255".

Valor de Retorno

- Con Éxitos Devuelve un arreglo : \$arreglo[1] conteniendo el socket real, \$arreglo[2] conteniendo la dirección IP y \$arreglo[3] conteniendo el puerto. Necesitaremos esta información en llamadas subsecuentes a UDPSend(), donde será pasado esta estructura/arreglo del socket.
- Al Fallar Devuelve \$array[0]=0 y establece @error a:
@error: valor de retorno de windows API WSAGetError(ver [MSDN](#)).

Comentarios

Esta función es utilizada por el cliente para comunicarse con el servidor.

Relativo

[UDPBInd](#), [UDPSend](#), [UDPCloseSocket](#), [UDPRecv](#)

Ejemplo

```
;CLIENTE!!!!!!! Iniciar el SERVIDOR Primero... marioneta !!
$g_IP = "127.0.0.1"
```

```
; Iniciando los servicios UDP
=====
UDPStartUp()
```

```

; Conectando a un listening "SOCKET"
=====
$socket = UDPOpen( $g_IP, 65432 )
If @error <> 0 Then Exit

```

UDPRcv

Recibe datos de un socket abierto.

UDPRcv (socketarray, Longitud máxima [, flag])

Parámetros

socketarray	El socket/arreglo devuelto por la función UDPBnd .
Longitud máxima	# máximo de caracteres a recibir.
flag	<p>[opcional] Fuerza a la función a recibir datos binarios si es establecido a 1 (por defecto es 0, y detectará entre binario/cadena)</p> <p>Fuerza a la función a retornar desde IP/port si se establece a 2. El resultado es devuelto en un arreglo: [0] dato, [1] desde IP, [2] desde Puerto</p> <p>If you want both just use 3.</p>

Valor de Retorno

Con Éxitos: Devuelve un binario/cadena enviado por el socket abierto o un arreglo si flag = 2 o 3.

Al Fallar: Devuelve "" y establece @error a:

@error: -1, -2 or -3 socketarray inválido.

valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Por razones de compatibilidad esta función intentará devolver siempre una cadena. Si caracteres nulos (0x00) son recibidos entonces el valor devuelto será un tipo binario. Para forzar la función a devolver siempre un dato binario(La opción más sensible) utilice el parámetro "flag" a 1.

Relativo

[BinaryLen](#), [BinaryMid](#), [UDPBnd](#), [UDPOpen](#)

Ejemplo

```

;;Este es el servidor UDP
;;Iniciar esto primero
=====
; Iniciando los servicios UDP
=====
```

```

UDPStartup()
; Registra una función de limpieza
OnAutoItExitRegister("Cleanup")

; Conectarse a un SOCKET
=====
$socket = UDPBind("127.0.0.1", 65532)
If @error <> 0 Then Exit

While 1
    $data = UDPRecv($socket, 50)
    If $data <> "" Then
        MsgBox(0, "DATO UDP", $data, 1)
    Endif
    sleep(100)
WEnd

Func Cleanup()
    UDPCloseSocket($socket)
    UDPShutdown()
EndFunc

```

UDPSend

Envía un dato por un socket abierto.

UDPSend (socketarray, dato)

Parámetros

socketarray	El principal socket/arreglo devuelto por la función UDPOpen
dato	binario/cadena para ser enviada por el socket conectado.

Valor de Retorno

Con Éxitos Devuelve número de bytes enviados por el socket abierto.

Al Fallar Establece @error a:

@error: -1, -2 o -3 socketarray inválido.

1 IPAddr es incorrecto

2 port es incorrecto

valor de retorno de windows API WSAGetError (ver [MSDN](#)).

Comentarios

Ninguno.

Relativo

[UDPOpen](#), [UDPBind](#)

Ejemplo

```
;Este es el cliente UDP
;Iniciar el servidor primero

; Iniciando los servicios UDP
;=====
UDPStartup()
; Registra una función de limpieza
OnAutoItExitRegister("Cleanup")

; Abriendo el "SOCKET"
;=====
$socket = UDPOpen("127.0.0.1", 65532)
If @error <> 0 Then Exit

$n=0
While 1
    Sleep(2000)
    $n = $n + 1
    $status = UDPSend($socket, "Mensaje #" & $n)
    If $status = 0 then
        MsgBox(0, "ERROR", "Error mientras se enviaba el mensaje por UDP : " & @error)
        Exit
    EndIf
WEnd

Func Cleanup()
    UDPCloseSocket($socket)
    UDPShutdown()
EndFunc
```

Referencia de Referencia de Obj/COM

A continuación se muestra una lista completa de las funciones de Referencia de Obj/COM disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
ObjCreate	Crea una referencia a un objeto COM de el nombre de clase dado.
ObjEvent	Maneja eventos entrantes del Objeto dado.
ObjGet	Recupera una referencia a un objeto COM de un proceso existente o nombre de archivo.
ObjName	Devuelve el nombre o descripción de interfaz de un Objeto.

ObjCreate

Crea una referencia a un objeto COM de el nombre de clase dado.

***ObjCreate ("NombreDeClase" [, "NombreDeServidor" [, "NombreDeUsuario",
["contraseña"]]])***

Parámetros

NombreDeClase	La clase de el objeto en el siguiente formato: "Nombre de Aplicación.Tipo de Objeto"
NombreDeServidor	[opcional] nombre de un computador remoto del cual el objeto debe ser obtenido.
NombreDeUsuario	[opcional] nombre de el usuario al que la computadora remota tiene entrada como este formato "computadora\códigoDeUsuario" o "dominio\códigoDeUsuario".
Contraseña	[opcional] contraseña para el usuario en la computadora remota.

Valor de Retorno

Con Éxito: Devuelve un objeto.

Al Fallar: Devuelve 0 y establece el valor de @error en 1.

Comentarios

Use ObjCreate() si usted desea tener una nueva instancia de la aplicación referida. Si usted desea conectar a un proceso existente, use ObjGet().

Tome en cuenta que no todas las computadoras deben tener la misma forma de fijar Objetos. Así que **siempre** es mejor revisar para errores después de llamar a ObjCreate().

Los siguientes requerimientos aplican si usted desea acceder a objetos en computadoras *remotas*:

- El usuario ejecutando el script debe tener los permisos apropiados.
- El Objeto en el computador remoto debe soportar DCOM (COM Distribuido)
- El computador remoto debe tener 'Servicio de Registro Remoto' y servicio de 'compartir Archivo e Impresora' ejecutando.

Ver el [Obj/Referencia COM](#) para más información sobre Objetos.

Relativo

[GUICtrlCreateObj](#), [IsObj](#), [ObjEvent](#), [ObjGet](#), [ObjName](#)

Ejemplo

```
; Ejemplo 1
;
; Contando el número de ventanas abiertas shell

$oShell = ObjCreate("shell.application") ; Obtiene el Objeto de Shell de Windows
$oShellWindows=$oShell.windows ; Obtiene la colección de ventanas Shell abiertas

if Isobj($oShellWindows) then

    $string="" ; Cadena para propósitos de muestra

    for $Window in $oShellWindows ; Contando todas las ventanas existentes
        $String = $String & $Window.LocationName & @CRLF
    next

    MsgBox(0,"Shell Windows","Usted tiene las siguientes ventanas:" & @CRLF & @CRLF &
    $String);

endif
exit

; Ejemplo 2
;
; Abre el MediaPlayer en una computadora REMOTA
$oRemoteMedia = ObjCreate("MediaPlayer.MediaPlayer.1","nombre-de-computadora")

If not @error then
    MsgBox(0,"Prueba de ObjCreate Remoto","ObjCreate() de un Objeto Medioplayer remoto
    con éxito!")
    $oRemoteMedia.Open( @WindowsDir & "\media\Windows XP Startup.wav") ; Ejecuta un
    sonido si el archivo está presente
Else
    MsgBox(0,"Prueba de ObjCreate Remoto","Falló al abrir el objeto remoto. Código de error: "
```

```
& hex(@error,8))  
Endif
```

ObjEvent

Maneja eventos entrantes del Objeto dado.

ObjEvent (\$VariableObjeto, "PrefijoFunción" [, "nombre de interfaz"])
ObjEvent ("Autolt.Error" [, "nombre de función"])

Parámetros

\$VariableObjeto	Una variable conteniendo un Objeto del cual usted desea recibir eventos.
"PrefijoFunción"	El prefijo de las funciones que usted define al apuntador que recibirá eventos. El prefijo es añadido por el nombre de método de los Objetos.
"nombre de interfaz"	[opcional] nombre de la interfaz de Evento a utilizar. Nota: Debe ser soportado como saliente por el Objeto y debe ser de tipo DISPATCH.

Valor de Retorno

Con Éxito: Devuelve un objeto o un nombre de función.

Al Fallar: Devuelve "" y establece el valor de @error en 1.

Comentarios

El primer formato es usado para recibir Eventos de Objeto dado.

Para recibir un evento específico, se crea un nombre de función Autolt usando el prefijo añadido con el nombre de evento.

El segundo formato es usado para el manejo de Errores COM. Si cualquier error de COM ocurre, la función dada es llamada.

Si el segundo parámetro es omitido, le devolverá el nombre del error actual de la función proveniente, si se presenta.

Ver en [Obj/Referencia COM](#) para una explicación detallada.

Relativo

[ObjGet](#), [IsObj](#), [ObjCreate](#), [GUICtrlCreateObj](#)

Ejemplo

; Ejemplo de ObjEvent

```

ProgressOn("Ejemplo", "Cargando página... ")
$oIE=ObjCreate("InternetExplorer.Application.1") ; Crea una aplicación de Internet Explorer
$SinkObject=ObjEvent($oIE,"IEEvent_","DWebBrowserEvents2") ; Asignando eventos a UDFs
empezando con IEEvent_

; haciendo algunas actividades de la navegación
$oIE.Visible=1
$oIE.RegisterAsDropTarget = 1
$oIE.RegisterAsBrowser = 1
$oIE.Navigate( "http://www.AutoItScript.com/" )

sleep(3000) ; Darle tiempo a la carga de la página

$SinkObject=0 ; Deteniendo los eventos de IE
$oIE.Quit ; Quitar IE
$oIE=0
exit

; uno de muchas funciones de Internet Explorer

Func IEEVENT_ProgressChange($Progress,$ProgressMax)
$percent = Int( ($Progress * 100) / $ProgressMax )
If $percent >= 0 And $percent <= 100 Then
    ProgressSet ( $percent , $percent & " porcentaje.", "cargando página web" )
EndIf

EndFunc

Exit

; Ejemplo de COM Error Handler
; -------

$oIE=ObjCreate("InternetExplorer.Application.1") ; Crea una aplicación de Internet Explorer

Global $g_eventerror = 0 ; a ser revisado para conocer su ocurrió un error. Debe ser reiniciado
después de manejarse.

$MyError = ObjEvent("AutoIt.Error","MyErrFunc") ; Inicializar un manejador de error COM

$oIE.UnknownMethod ; Deliberadamente llama un método indefinido

If $g_eventerror then
    $g_eventerror = 0
    MsgBox (0,"AutoItCOM test","Prueba pasada: Hay un error número: " & @error)
Else
    MsgBox (0,"AutoItCOM test","Prueba fallida!")
Endif

Exit

; Este es mi error COM, personalizado

```

Func MyErrFunc()

```
Msgbox(0,"AutoItCOM Test","Se interceptó un error COM!" & @CRLF & @CRLF &_
"err.description is: " & @TAB & $oMyError.description & @CRLF &_
"err.windescription:" & @TAB & $oMyError.windescription & @CRLF &_
"err.number is: " & @TAB & hex($oMyError.number,8) & @CRLF &_
"err.lastdllerror is: " & @TAB & $oMyError.lastdllerror & @CRLF &_
"err.scriptline is: " & @TAB & $oMyError.scriptline & @CRLF &_
"err.source is: " & @TAB & $oMyError.source & @CRLF &_
"err.helpfile is: " & @TAB & $oMyError.helpfile & @CRLF &_
"err.helpcontext is: " & @TAB & $oMyError.helpcontext_
)

Local $err = $oMyError.number
If $err = 0 Then $err = -1

$g_eventerror = $err ; para revisar luego que esta función devuelva
Endfunc
```

ObjGet

Recupera una referencia a un objeto COM de un proceso existente o nombre de archivo.

ObjGet ("NombreDeArchivo" [, "NombreDeClase"])

Parámetros

NombreDeArchivo	La ruta completa y nombre del archivo conteniendo el objeto.(Ver Comentarios)
NombreDeClase	(opcional) La clase del objeto en el siguiente formato: "Nombre de aplicación.Tipo de Objeto"

Valor de Retorno

Con Éxito: Devuelve un objeto.

Al Fallar: Devuelve 0 y establece el valor de @error en 1.

Comentarios

El nombre de archivo es opcional si usted desea usar solamente el nombre de la clase, pero el parámetro no puede ser omitido. Use una cadena vacía si usted solamente desea usar el nombre de la clase. Como: \$Objeto = ObjGet("", "Excel.Aplicación")

Si usted desea usar un nombre de archivo, el nombre de la clase es opcional. Es solamente requerido cuando hay múltiple clases definidas al mismo tipo de archivo y usted desea acceder a la clase específica.

Tome en cuenta que no todas las computadoras tienen la misma manera de fijar Objetos. Así que **siempre** es bueno revisar los errores después de llamar ObjGet().

Ver [Obj/Referencia COM](#) para más información sobre Objetos.

Relativo

[GUICtrlCreateObj](#), [IsObj](#), [ObjCreate](#), [ObjEvent](#), [ObjName](#)

Ejemplo

```
; Ejemplo obteniendo un Objeto usando el nombre de clase
;
; Excel debe ser activado para que este ejemplo sea realizado con éxito

$oExcel = ObjGet("""", "Excel.Application") ; Obteniendo el Objeto de Excel

if @error then
    MsgBox (0, "ExcelTest", "Error obteniendo un objeto activo de Excel. Código de Error: " &
hex(@error,8))
    exit
endif

$oExcel.Visible = 1 ; Que se muestre
$oExcel.workbooks.add ; Agregar nueva hoja de trabajo
exit

; Ejemplo obteniendo un Objeto usando un nombre de archivo
;
; Un archivo de Excel con el nombre de archivo Worksheet.xls debe ser creado en el directorio
raíz
; de la unidad C:\ para que este ejemplo trabaje.

$FileName="C:\Worksheet.xls"

if not FileExists($FileName) then
    MsgBox (0, "Excel File Test", "No se puede ejecutar esta prueba, porque no se creó el archivo de
Excel "& $FileName)
    Exit
endif

$oExcelDoc = ObjGet($FileName) ; Obteniendo el Objeto de Excel de un archivo existente

if IsObj($oExcelDoc) then

    ; Consejo: Para hacer visible Excel puede quitar el ";" de las siguientes líneas (crédito:
    DaleHohm)
    ; $oExcelDoc.Windows(1).Visible = 1; Establece la primer hoja de trabajo visible
    ; $oExcelDoc.Application.Visible = 1; Establece que la aplicación esté visible (sin esto Excel
    cerrará)
```

```

$String = "" ; Cadena para propósitos de muestra

; Algunas propiedades de documento no regresan un valor, ignoraremos estos en este caso.
$OEvent=ObjEvent("Autolt.Error","nada"); Igual al Error de VBscript

For $Property In $oExcelDoc.BuiltinDocumentProperties
    $String = $String & $Property.Name & ":" & $Property.Value & @CRLF
Next

Msgbox(0,"Excel File Test","Las propiedades del documento de " & $FileName & " son:" &
@CRLF & @CRLF & $String)

$oExcelDoc.Close ; Cerrar el documento de Excel

else
    MsgBox (0,"Excel File Test","Error: No se puede abrir "& $FileName & " como un Objeto de
Excel.")
endif

```

ObjName

Devuelve el nombre o descripción de interfaz de un Objeto.

ObjName (\$VariableDeObjeto [,Flag])

Parámetros

\$VariableDeObjeto	Una variable conteniendo un Objeto del nombre que usted desea recibir.
Bandera	[opcional] 1 = Nombre de la interfaz de envío siendo usado (por defecto) 2 = Descripción de el (documento) Objeto 3 = El ID de Programa del Objeto 4 = El DLL archivo del Objeto que fue creado (si es soportado) 5 = Nombre de archivo y posición de el ícono de caja de herramientas (si es soportado)

Valor de Retorno

Con Éxito: Devuelve a cadena representando el nombre.

Al Fallar: Establece @error y devuelve ""

Comentarios

No todos los Objetos soportan Flags 2 a 5. Siempre es bueno probar por medio de @error en estos casos.

Relativo

IsObj, ObjGet, ObjCreate

Ejemplo

```
$oInternet = ObjCreate("InternetExplorer.Application")
$oInternet.Navigate( "http://www.google.com" ) ; Abriendo una página web que contiene un formulario
sleep(4000) ; Dando tiempo a la página que cargue

$oDoc = $oInternet.document ; Ejemplo de objeto a probar
$oForm = $oDoc.forms(0) ; Ejemplo de objeto a probar

msgbox(0,"","Nombre de Interfaz de $oInternet es: " & ObjName($oInternet) & @CRLF &_
"Nombre de Objeto de $oInternet es: " & ObjName($oInternet,2) & @CRLF &_
"Nombre de Interfaz de $oDoc es: " & ObjName($oDoc) & @CRLF &_
"Nombre de Objeto de $oDoc es: " & ObjName($oDoc,2) & @CRLF &_
"Nombre de Interfaz de $oForm es: " & ObjName($oForm) & @CRLF &_
"Nombre de Objeto de $oForm es: " & ObjName($oForm,2))
```

Referencia de Administración de Procesos

A continuación se muestra una lista completa de las funciones de Administración de Procesos disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
DllCall	Invocación dinámica de una función en una DLL.
DllCallbackFree	Libera un apuntador creado previamente con DllCallbackRegister.
DllCallbackGetPtr	Devuelve el puntero para una función callback que puede ser pasado al API Win32.
DllCallbackRegister	Crea una función DLL Callback definida por usuario.
DllClose	Cierra un DLL previamente abierto.
DllOpen	Abre un archivo DLL para usar en DllCall.
DllStructCreate	Crea una estructura en estilo C/C++ para ser usado en DllCall.
DllStructGetData	Devuelve el dato de un elemento de la estructura.
DllStructGetPtr	Devuelve el apuntador de la estructura o del elemento dentro de la estructura.
DllStructGetSize	Devuelve el tamaño de una estructura DLL en bytes.
DllStructSetData	Establece el dato en uno de los elementos en la estructura.
PluginOpen	Abre un fichero de plugin.
PluginClose	Cerrar un fichero plugin.
ProcessClose	Termina un proceso nombrado.
ProcessExists	Revisa si un proceso especificado existe.
ProcessGetStats	Devuelve un arreglo sobre Memoria o información de IO(Entrada y Salida) de un proceso en ejecución.
ProcessSetPriority	Cambia la prioridad de un proceso
ProcessList	Devuelve un arreglo listando los procesos actualmente en ejecución (nombres y PIDs)
ProcessWait	Pausa el script ejecución hasta que proceso dado exista.
ProcessWaitClose	Pausa el script en ejecución hasta que dado proceso termine.
Run	Ejecuta un programa externo.
RunWait	Ejecuta un programa externo y pausa la ejecución del script hasta que

	el programa finaliza.
RunAs	Ejecuta un programa externo bajo el contexto de un usuario diferente.
RunAsWait	Ejecuta un programa externo bajo el contexto de un usuario diferente y pausa la ejecución del script hasta que el programa finaliza.
ShellExecute	Ejecuta un programa externo usando la API de ShellExecute.
ShellExecuteWait	Ejecuta un programa externo usando la API de ShellExecute y pausa la ejecución del script hasta que finalice.
Shutdown	Apaga el sistema.
StderrRead	Lee del flujo de STDERR de un proceso hijo previamente ejecutado.
StdinWrite	Escribe un número de caracteres al flujo de STDIN de un proceso hijo previamente ejecutado.
StdioClose	Cierra todos los recursos asociados con un proceso previamente ejecutado redirección de STDIO.
StdoutRead	Lee del flujo STDOUT de un proceso hijo previamente ejecutado.

DllCall

Invocación dinámica de una función en una DLL.

DllCall ("dll", "tipo de retorno", "función" [, tipo1, parámetro1 [, tipo n, parámetro n]])

Parámetros

dll	El nombre del archivo DLL a usar. Eje. "user32.dll". Un apuntador obtenido por DllOpen puede también ser usado (Ver observaciones).
tipo de retorno	El tipo de dato que retorna la función (Ver más abajo).
función	El nombre, ejemplo. "MessageBox" o el valor ordinal, Eje. 62, de la función en la DLL a invocar.
tipo1	[opcional] El tipo de dato del parámetro (ver observaciones).
parámetro1	[opcional] El parámetro actual (ver observaciones).
tipo n	[opcional] El tipo de enésimo parámetro (ver comentarios).
parámetro n	[opcional] El parámetro enésimo actual (ver comentarios).

Los tipos de datos válidos son:

Tipo	Detalles
none	Sin valor (Solo válido para tipo return- equivalente al vacío en C)
BYTE	un entero de 8 bit sin signo

BOOLEAN	un entero de 8 bit sin signo
short	un entero de 16 bit
USHORT	un entero de 16 bit sin signo
WORD	un entero de 16 bit sin signo
int	un entero de 32 bit
long	un entero de 32 bit
BOOL	un entero de 32 bit
UINT	un entero de 32 bit sin signo
ULONG	un entero de 32 bit sin signo
DWORD	un entero de 32 bit sin signo
INT64	un entero de 64 bit
UINT64	un entero de 64 bit sin signo
ptr	un puntero general(void *)
HWND	un puntero de ventana (pointer)
HANDLE	un identificador (pointer)
float	un número de punto flotante de presión simple.
double	un número de punto flotante de presión doble.
INT_PTR, LONG_PTR, LRESULT, LPARAM	un entero lo suficientemente grande para mantener un puntero cuando se corre bajo versiones x86 o x64 de Autolt.
UINT_PTR, ULONG_PTR, DWORD_PTR, WPARAM	un entero sin signo lo suficientemente grande para mantener un puntero cuando se corre bajo versiones x86 o x64 de Autolt.
str	una cadena ANSI (un mínimo de 65536 caracteres es asignado).
wstr	una cadena extensa de caracteres UNICODE (a minimum of 65536 chars is allocated).
*	Adicionar * al final de otro tipo de dato para pasarlo por referencia. Por ejemplo "int*" pasa un puntero a un tipo "int".

Conversiones de tipos de API de Windows para tipos de Autolt:

WINDOWS API Type	Autolt Type
LPCSTR/LPSTR	str
LPCWSTR/LPWSTR	wstr
LPVOID	ptr
LPxyz	xyz*
HINSTANCE	handle

HRESULT	long
LONGLONG/LARGE_INTEGER	INT64
ULLONG/ULARGE_INTEGER	UINT64
SIZE_T	ULONG_PTR

Valor de Retorno

- Con Éxitos @error = 0.
- Al Fallar: establece @error
- @error:
- 1 incapaz de usar el fichero DLL,
 - 2 "tipo de retorno" desconocido,
 - 3 "función" no encontrada en el fichero DLL,
 - 4 número de parámetros inválido.

Ver observaciones.

Comentarios

Si el nombre de un archivo dll es dado entonces el DLL es automáticamente cargado y cerrado al finalizar este. Si usted desea un control manual de la carga y descarga del archivo DLL entonces usted debe usar DllOpen y DllClose usando un apuntador en lugar del nombre del archivo en esta función.

Por defecto, AutoIt usa el método de llamada 'stdcall'. Para usar el método 'cdecl' colocar ':cdecl' después del "tipo de retorno".

```
DllCall("SQLite.dll", "int:cdecl", "sqlite3_open", "str", $sDatabase_Filename , "long*", 0).
```

Por defecto, AutoIt intenta usar la versión ANSI del nombre de una función, ej. MessageBoxA es procurado cuando MessageBox es tomado como nombre de la función. Para realizar una invocación en versión unicode use MessageBoxW.

Si la invocación de la función falla entonces @error es fijado a 1. De otro modo un arreglo es devuelto conteniendo el valor de la función y una copia de todos los parámetros (incluyendo parámetros que la función pueda tener modificados cuando pase por referencia).

```
$return[0] = valor de retorno de la función
$return[1] = parámetro1
$return[2] = parámetro2
...
$return[n] = parámetro n
```

Relativo

[DllCallbackFree](#), [DllCallbackGetPtr](#), [DllCallbackRegister](#), [DllOpen](#), [DllClose](#), [DllStructCreate](#), [DllStructGetPtr](#)

Ejemplo

```
; ****
; Ejemplo 1 - invocando la API MessageBox directamente
; ****

$result = DllCall("user32.dll", "int", "MessageBox", "hwnd", 0, "str", "Some text", "str", "Some
title", "int", 0)

;

; ****
; Ejemplo 2 - invocando una función que modifica parámetros
; ****

$hwnd = WinGetHandle("[CLASS:Notepad]")
$result = DllCall("user32.dll", "int", "GetWindowText", "hwnd", $hwnd, "str", "", "int", 32768)
$msgbox(0, "", $result[0]) ; números de caracteres devueltos
$msgbox(0, "", $result[2]) ; Texto devuelto en parámetro 2

;

; ****
; Ejemplo 3 - Mostrar la ventana PickIconDlg
; ****

$fileName = @SystemDir & '\shell32.dll'

; Creando una estructura para almacenar el índice de iconos
$stIcon = DllStructCreate("int")

$stString = DLLStructCreate("wchar[260]")
$structsize = DllStructGetSize($stString)/2
DllStructSetData($stString, 1, $fileName)

; ejecutando PickIconDlg - '62' es el valor ordinal para esta función
DllCall("shell32.dll", "none", 62, "hwnd", 0, "ptr", DllStructGetPtr($stString), "int", $structsize,
"ptr", DllStructGetPtr($stIcon))

$fileName = DllStructGetData($stString, 1)
$iconIndex = DllStructGetData($stIcon, 1)

; Mostrando el nuevo fichero y ícono indexado
Msgbox(0, "Info", "Last selected file: " & $fileName & @LF & "Icon-Index: " & $iconIndex)
```

DllCallbackFree

Libera un apuntador creado previamente con DllCallbackRegister.

DllCallbackFree (*apuntador*)

Parámetros

apuntador	El apuntador DllCallback, devuelto por una llamada previa a DllCallbackRegister.
-----------	--

Valor de Retorno

Ninguno.

Comentarios

Antes de finalizar, Autolt automáticamente cierra cualquier apuntador abierto pero la invocación de DllCallbackFree sigue siendo una buena idea!

Relativo

[DllCall](#), [DllCallbackGetPtr](#), [DllCallbackRegister](#)

Ejemplo

```
; Crea la función callback
$handle = DLLCallbackRegister ("_EnumWindowsProc", "int", "hwnd;Iparam")

; Invoca EnumWindows
DllCall("user32.dll", "int", "EnumWindows", "ptr", DllCallbackGetPtr($handle), "Iparam", 10)

; Borra la función callback
DllCallbackFree($handle)

; Prodecimiento Callback
Func _EnumWindowsProc($hWnd, $iParam)
    If WinGetTitle($hWnd) <> "" And BitAnd(WinGetState($hWnd), 2) Then
        $res = MsgBox(1, WinGetTitle($hWnd), "$hWnd=" & $hWnd & @CRLF & "IParam=" &
        $iParam & @CRLF & "$hWnd(type)=" & VarGetType($hWnd))
        If $res = 2 Then Return 0 ; Calcelar clickeado, devuelve 0 para detener para enumeración
    EndIf
    Return 1 ; Devuelve 1 y continua la enumeración
EndFunc
```

DllCallbackGetPtr

Devuelve el puntero para una función callback que puede ser pasado al API Win32.

DllCallbackGetPtr (apuntador)

Parámetros

apuntador	Un apuntador DllCallback devuelto a partir de DllCallbackRegister.
-----------	--

Valor de Retorno

Con Éxitos El puntero para la función callback.

Al Fallar 0.

Comentarios

Use DllCallbackGetPtr para pasar la dirección de una función callback para la API Win32 cuando usamos DllCall.

Relativo

[DllCall](#), [DllCallBackRegister](#), [DllCallbackFree](#)

Ejemplo

```
; Crea la función callback
$handle = DLLCallbackRegister ("_EnumWindowsProc", "int", "hwnd;Iparam")

; Invoca EnumWindows
DllCall("user32.dll", "int", "EnumWindows", "ptr", DllCallbackGetPtr($handle), "Iparam", 10)

; Borra la función callback
DllCallbackFree($handle)

; Prodecimiento Callback
Func _EnumWindowsProc($hWnd, $iParam)
    If WinGetTitle($hWnd) <> "" And BitAnd(WinGetState($hWnd), 2) Then
        $res = MsgBox(1, WinGetTitle($hWnd), "$hWnd=" & $hWnd & @CRLF & "IParam=" &
        $iParam & @CRLF & "$hWnd(type)=" & VarGetType($hWnd))
        If $res = 2 Then Return 0 ; Calcelar clickeado, devuelve 0 para detener para enumeración
    Endif
    Return 1 ; Devuelve 1 y continua la enumeración
EndFunc
```

DllCallbackRegister

Crea una función DLL Callback definida por usuario.

DllCallbackRegister ("función", "tipo de retorno", "parámetros")

Parámetros

función	El nombre de la Función Definida de Usuario a invocar.
tipo de retorno	El tipo de dato de retorno y la convención de invocación de la función (ver DllCall).
parámetros	Una lista de parámetros separados por ";" que será pasado a la función. Use los tipos de datos DllCall.

Valor de Retorno

Con Éxitos	Devuelve un apuntador "handle" dll para ser usado con DllCallbackGetPtr y funciones DllCallbackFree.
Al Fallar	Devuelve 0 si ocurre un error.

Comentarios

Cuando finaliza un trabajo con callback, invoque la función DllCallbackFree para cerrar este. Autolt normalmente cierra todos los archivos al finalizar, pero el uso de DllCallbackFree es una buena idea.

Relativo

[DllCall](#), [DllCallbackFree](#), [DllCallbackGetPtr](#)

Ejemplo

```
; Crea una función callback
$handle = DLLCallbackRegister ("_EnumWindowsProc", "int", "hwnd;Iparam")

; Invoca EnumWindows
DllCall("user32.dll", "int", "EnumWindows", "ptr", DllCallbackGetPtr($handle), "Iparam", 10)

; Borra la función callback
DllCallbackFree($handle)

; Procedimiento Callback
Func _EnumWindowsProc($hWnd, $iParam)
    If WinGetTitle($hWnd) <> "" And BitAnd(WinGetState($hWnd), 2) Then
        $res = MsgBox(1, WinGetTitle($hWnd), "$hWnd=" & $hWnd & @CRLF & "IParam=" &
        $iParam & @CRLF & "$hWnd(type)=" & VarGetType($hWnd))
        If $res = 2 Then Return 0 ; Calcelar clickeado, devuelve 0 y detiene la enumeración
    EndIf
    Return 1 ; Devuelve 1 y continua la enumeración
EndFunc
```

DllClose

Cierra un DLL previamente abierto.

DllClose (dllhandle)

Parámetros

dllhandle	El apuntador de un dll, devuelto por una llamada previa de DllOpen.
-----------	---

Valor de Retorno

Ninguno.

Comentarios

Antes de finalizar, Autolt automáticamente cierra cualquier dlls abierta, pero invocar DllClose es una buena idea!

Relativo

[DllCall](#), [DllOpen](#)

Ejemplo

```
$dll = DllOpen("user32.dll")
$result = DllCall($dll, "int", "MessageBox", "hwnd", 0, "str", "algún texto", "str", "algún título",
"int", 0)
DllClose($dll)
```

DllOpen

Abre un archivo DLL para usar en DllCall.

DllOpen ("Nombre de fichero")

Parámetros

Nombre de fichero	Nombre de un fichero DLL para ser abierto.
-------------------	--

Valor de Retorno

Con Éxitos Devuelve un apuntador "handle" dll para ser usado con funciones Dll.
Al Fallar Devuelve -1 si ocurre un error.

Comentarios

Cuando finaliza un trabajo con una dll, invoque la función DllClose para cerrar este. Autolt normalmente cierra todos los archivos al finalizar, pero el uso de DllClose es una buena idea.

Relativo

[DllCall](#), [DllClose](#)

Ejemplo

```
$dll = DllOpen("user32.dll")
$result = DllCall($dll, "int", "MessageBox", "hwnd", 0, "str", "algún texto", "str", "algún título",
"int", 0)
DllClose($dll)
```

DllStructCreate

Crea una estructura en estilo C/C++ para ser usado en DllCall.

DllStructCreate ("Estructura" [,Puntero])

Parámetros

"Estructura"	Una cadena que representa la estructura a crear (Ver Observaciones).
Puntero	[opcional] Si es suplido la estructura no asigna memoria pero usa el puntero dado.

Valor de Retorno

Con Éxitos	Una variable para ser usado con DllStruct calls.
Al Fallar	0.
@Error:	0 = No Error. 1 = Variable pasado a DllStructCreate no es una cadena. 2 = Resulta un tipo de dato desconocido en la cadena pasada. 3 = Falla la asignación de memoria necesaria para la estructura, o el puntero = 0. 4 = Error asignando memoria para la cadena pasada.

Type	Details
BYTE	8bit(1byte) carácter con signo
BOOLEAN	8bit(1byte) carácter sin signo
CHAR	8bit(1byte) ASCII char
WCHAR	16bit(2byte) carácter UNICODE extenso
short	16bit(2bytes) entero con signo
USHORT	16bit(2bytes) entero sin signo
WORD	16bit(2bytes) entero sin signo
int	32bit(4bytes) entero con signo
long	32bit(4bytes) entero con signo
BOOL	32bit(4bytes) entero con signo
UINT	32bit(4bytes) entero sin signo
ULONG	32bit(4bytes) entero sin signo
DWORD	32bit(4bytes) entero sin signo
INT64	64bit(8bytes) entero con signo
UINT64	64bit(8bytes) entero sin signo
ptr	32 or 64bit entero sin signo (dependiendo si es usado la versión x86 o x64 version de AutoIt)

HWND	32bit(4bytes) entero
HANDLE	32bit(4bytes) entero
float	32bit(4bytes) punto flotante
double	64bit(8bytes) punto flotante
INT_PTR, LONG_PTR, LRESULT, LPARAM	32 or 64bit entero con signo (dependiendo si es usado la versión x86 o x64 version de Autolt)
UINT_PTR, ULONG_PTR, DWORD_PTR, WPARAM	32 or 64bit entero sin signo (dependiendo si es usado la versión x86 o x64 version de Autolt)

Comentarios

Cada tipo de dato debe ser separado por un ';'. Cree arreglos por adición del '[tamaño]' después del tipo de dato.

ejemplo:`DllStructCreate("int;char[128]");`

`DllStructCreate("byte;double")` ; la estructura es de 16 bytes, el "double" es un desplazamiento de 8.

`DllStructCreate("align 4;byte;double")` ; la estructura es de 12 bytes, el "double" es un desplazamiento de 4.

Si es necesario un cambio en la alineación, "align" puede ser adicionado antes del primer elemento que necesita ser cambiado.

"align" o "align 8" representa el alineamiento por defecto.

Para liberar memoria asignada simplemente establezca la variable de retorno a 0.

Relativo

[DllCall](#), [DllStructGetData](#), [DllStructSetData](#), [DllStructGetPtr](#), [DllStructGetSize](#), [IsDllStruct](#)

Ejemplo

```
;=====
; Crea la estructura
; struct {
;   int      var1;
;   unsigned char var2;
;   unsigned int var3;
;   char     var4[128];
; }
```

=====

```
$str    = "int var1;ubyte var2;uint var3;char var4[128]"
```

```

$a      = DllStructCreate($str)
if @error Then
    MsgBox(0,"","Error in DllStructCreate " & @error);
    exit
endif

;=====;
; Establece datos en la estructura
; struct.var1 = -1;
; struct.var2 = 255;
; struct.var3 = INT_MAX; -1 will be typecasted to (unsigned int)
; strcpy(struct.var4,"Hola");
; struct.var4[0] = 'h';
;=====;
DllStructSetData($a,"var1",-1)
DllStructSetData($a,"var2",255)
DllStructSetData($a,"var3",-1)
DllStructSetData($a,"var4","Hola")
DllStructSetData($a,"var4",Asc("h"),1)

;=====;
; Mostrando la información de la estructura
;=====;
MsgBox(0,"DllStruct","Tamaño de estructura: " & DllStructGetSize($a) & @CRLF & _
    "Puntero de estructura: " & DllStructGetPtr($a) & @CRLF & _
    "Dato:" & @CRLF & _
    DllStructGetData($a,1) & @CRLF & _
    DllStructGetData($a,2) & @CRLF & _
    DllStructGetData($a,3) & @CRLF & _
    DllStructGetData($a,4))

;=====;
; Liberando memoria asignada a la estructura
;=====;
$a=0

```

DllStructGetData

Devuelve el dato de un elemento de la estructura.

DllStructGetData (Estructura, Elemento [, índice])

Parámetros

Estructura	La estructura devuelta por DllStructCreate.
Elemento	El elemento de la estructura a acceder, Iniciando en 1 o el nombre de la estructura dado en DllStructCreate.

índice	[opcional] Para elementos que son arreglos esto especifica el índice de base-cero a recibir. Si es omitida la palabra reservada Default keyword entonces el arreglo entero es devuelto (muy útil para recibir cadenas). No usado para elementos que no sean arreglos.
--------	---

*char[n], byte[n] y ubyte[n] devuelve todos los datos de elementos cuando índice es omitido.

Valor de Retorno

Con Éxitos	Dato en el elemento de la estructura.
Al Fallar	0.
@Error:	0 = No Error. 1 = La estructura no es una estructura correcta creada por DllStructCreate. 2 = El valor de elemento esta fuera de rango. 3 = índice fuera de la estructura. 4 = El tipo de dato del elemento es desconocido. 5 = índice <= 0.

Comentarios

Cuando el elemento es char[n] y índice es omitido el dato devuelto es una Cadena. Cuando el elemento es byte[n] o ubyte[n] y índice es omitido el dato devuelto es tipo binario, de otro modo siempre devuelve un número.

Relativo

[DllStructCreate](#), [DllStructSetData](#)

Ejemplo

```
$p = DllStructCreate("dword dwOSVersionInfoSize;dword dwMajorVersion;dword
dwMinorVersion;dword dwBuildNumber;dword dwPlatformId;char szCSDVersion[128]")

;Pensar en esto como p->dwOSVersionInfoSize = sizeof(OSVERSIONINFO)
DllStructSetData($p, "dwOSVersionInfoSize", DllStructGetSize($p))

;haciendo el DllCall
$ret = DllCall("kernel32.dll","int","GetVersionEx","ptr",DllStructGetPtr($p))

;if Not $ret[0] Then
  MsgBox(0,"DllCall Error","DllCall Failed")
  exit
EndIf

;obtener los valore de retorno
$major    = DllStructGetData($p,"dwMajorVersion")
$minor    = DllStructGetData($p,"dwMinorVersion")
$build    = DllStructGetData($p,"dwBuildNumber")
$platform = DllStructGetData($p,"dwPlatformId")
$version  = DllStructGetData($p,"szCSDVersion")
```

```

;liberando la estructura
$p =0

msgbox(0,"","Mayor: " & $major & @CRLF &_
"Menor: " & $minor & @CRLF &_
"Build: " & $build & @CRLF &_
"ID plataforma: " & $platform & @CRLF &_
"Versión: " & $version)

```

DllStructGetPtr

Devuelve el apuntador de la estructura o del elemento dentro de la estructura.

DllStructGetPtr (estructura [,elemento])

Parámetros

estructura	La estructura devuelta por DllStructCreate.
elemento	(opcional) El elemento de la estructura cuyo apuntador usted necesita, inicia en 1 o el nombre del elemento definido con DllStructCreate.

Valor de Retorno

Con Éxitos: El apuntador de la estructura
 Al Fallar: 0.
 @Error: 0 = Sin error.
 1 = Estructura incorrecta según esta definida por DllStructCreate.
 2 = Elemento fuera de límite.

Comentarios

Usado en DllCall.

Relativo

[DllCall](#), [DllStructCreate](#)

Ejemplo

```

;Ejemplo1
;obtener el identificador de ventana y usando WinGetPos para obtener el rectángulo de
ventana
$hwnd = WinGetHandle("")
$coor = WinGetPos($hwnd)

```

```

;crea la estructura
$rect = DllStructCreate("int;int;int;int")

;hacer el DllCall
DLLCall("user32.dll","int","GetWindowRect",
        "hwnd",$hwnd,
        "ptr",DllStructGetPtr($rect)) ; use DllStructGetPtr when calling DllCall

;obteniendo el rectángulo devuelto
$I = DllStructGetData($rect,1)
$t = DllStructGetData($rect,2)
$r = DllStructGetData($rect,3)
$b = DllStructGetData($rect,4)

;Liberando la estructura
$rect = 0

;mostrando el resultado de WinGetPos y el rectángulo retornado
MsgBox(0,"The Larry Test :)","WinGetPos(): (" & $coor[0] & "," & $coor[1] &_
      ") (" & $coor[2] + $coor[0] & "," & $coor[3] + $coor[1] & ")" & @CRLF &_
      "GetWindowRect(): (" & $I & "," & $t & ") (" & $r & "," & $b & ")")

;Ejemplo2
; DllStructGetPtr refiereñando un item
$a      = DllStructCreate("int")
if @error Then
  MsgBox(0,"","Error en DllStructCreate " & @error);
  exit
endif

$b = DllStructCreate("uint",DllStructGetPtr($a,1))
if @error Then
  MsgBox(0,"","Error en DllStructCreate " & @error);
  exit
endif

$c = DllStructCreate("float",DllStructGetPtr($a,1))
if @error Then
  MsgBox(0,"","Error en DllStructCreate " & @error);
  exit
endif

;estableciendo el dato
DllStructSetData($a,1,-1)

;=====
; Mostrando los diferentes tipos de datos en algunos datos
;=====

MsgBox(0,"DllStruct",_
      "int: " & DllStructGetData($a,1) & @CRLF &_
      "uint: " & DllStructGetData($b,1) & @CRLF &_
      "float: " & DllStructGetData($c,1) & @CRLF &_

```

```
"")

; liberando memoria asignada
$a = 0
```

DllStructGetSize

Devuelve el tamaño de una estructura DLL en bytes.

DllStructGetSize (Estructura)

Parámetros

Estructura	La estructura devuelta por DllStructCreate.
------------	---

Valor de Retorno

Con Éxitos: El tamaño de una estructura en bytes.

Al Fallar: 0.

@Error: 0 = Sin error.

1 = Estructura incorrecta según esta definida por DllStructCreate.

Comentarios

Usado cuando el dato en la estructura necesita tener el tamaño de la estructura.

Relativo

[DllStructCreate](#), [DllStructSetData](#)

Ejemplo

```
;=====
; Crea la estructura
; struct {
;   int      var1;
;   unsigned char var2;
;   unsigned int var3;
;   char     var4[128];
; }
;=====
$str    = "int;ubyte:uint;char[128]"
$a      = DllStructCreate($str)
if @error Then
  MsgBox(0,"","Error en DllStructCreate " & @error);
  exit
endif
```

```

;=====
; Establece datos en la estructura
; struct.var1 = -1;
; struct.var2 = 255;
; struct.var3 = INT_MAX; -1 will be typecasted to (unsigned int)
; strcpy(struct.var4,"Hola");
; struct.var4[0] = 'h';
;=====

DllStructSetData($a,1,-1)
DllStructSetData($a,2,255)
DllStructSetData($a,3,-1)
DllStructSetData($a,4,"Hello")
DllStructSetData($a,4,Asc("h"),1)

;=====
; Mostrando la información de la estructura
;=====

MsgBox(0,"DllStruct","Struct Size: " & DllStructGetSize($a) & @CRLF &_
"Struct pointer: " & DllStructGetPtr($a) & @CRLF &_
>Data:" & @CRLF &_
DllStructGetData($a,1) & @CRLF &_
DllStructGetData($a,2) & @CRLF &_
DllStructGetData($a,3) & @CRLF &_
DllStructGetData($a,4))

;=====
; Liberando memoria asignada a la estructura
;=====

$a = 0

```

DllStructSetData

Establece el dato en uno de los elementos en la estructura.

DllStructSetData (Estructura, Elemento, valor [, índice])

Parámetros

Estructura	La estructura devuelta por DllStructCreate.
Elemento	El elemento de la estructura que usted desea acceder, inicia en 1 o el nombre del elemento según lo definido por DllStructCreate.
valor	El nuevo valor para colocar en la estructura
índice	[opcional] Si el elemento es un arreglo, usted necesita especificar qué índice va a establecer, de otro modo se toma el índice 1. El primer elemento es 1.

Valor de Retorno

Con Éxitos: El valor, leída de la estructura.

Al Fallar: 0.
 @Error: 0 = sin error.
 1 = Estructura incorrecta según esta definida por DllStructCreate.
 2 = Valor del elemento fuera de rango.
 3 = índice sobrepasa la estructura.
 4 = El tipo de dato del elemento es desconocido
 5 = índice <= 0.

Comentarios

Cuando el elemento es char[n], byte[n] o ubyte[n] el dato puede ser cadena, de otro modo es tomado como número.

Relativo

[DllStructCreate](#), [DllStructGetData](#), [DllStructGetSize](#)

Ejemplo

```
=====
; Crea la estructura
; struct {
;   int      var1;
;   unsigned char  var2;
;   unsigned int   var3;
;   char       var4[128];
; }
=====
$str    = "int;ubyte:uint;char[128]"
$a     = DllStructCreate($str)
if @error Then
  MsgBox(0,"","Error en DllStructCreate " & @error);
  exit
endif

=====
; Establece datos en la estructura
; struct.var1 = -1;
; struct.var2 = 255;
; struct.var3 = INT_MAX; -1 will be typecasted to (unsigned int)
; strcpy(struct.var4,"Hola");
; struct.var4[0] = 'h';
=====
DllStructSetData($a,1,-1)
DllStructSetData($a,2,255)
DllStructSetData($a,3,-1)
DllStructSetData($a,4,"Hello")
DllStructSetData($a,4,Asc("h"),1)
=====
; Mostrando la información de la estructura
```

```

;=====
MsgBox(0,"DllStruct","Struct Size: " & DllStructGetSize($a) & @CRLF &_
"Struct pointer: " & DllStructGetPtr($a) & @CRLF &_
>Data:" & @CRLF &_
DllStructGetData($a,1) & @CRLF &_
DllStructGetData($a,2) & @CRLF &_
DllStructGetData($a,3) & @CRLF &_
DllStructGetData($a,4))

;=====
; Liberando memoria asignada a la estructura
;=====
$a = 0

```

PluginOpen

Abre un fichero de plugin.

PluginOpen (nombre_fichero)

Aviso: Esta característica es experimental. Puede no funcionar, puede contener errores, ser cambiado o removido sin aviso alguno.

No reporte ningún fallo o recomiende nuevas ideas acerca de esta característica.
USAR BAJO SU PROPIO RIESGO.

Parámetros

nombre_fichero	El nombre del fichero de plugin para abrir.
----------------	---

Valor de Retorno

Con Éxitos: Devuelve un "apuntador" dll para poder cerrarse con PluginClose().

Al Fallar: Devuelve 0.

Comentarios

Ninguno.

Relativo

[PluginClose](#)

Ejemplo

`$handle = PluginOpen("ejemplo.dll")`

PluginClose

Cerrar un fichero plugin.

PluginClose (apuntador_dll)

Aviso: Esta característica es experimental. Puede no funcionar, puede contener errores, ser cambiado o removido sin aviso alguno.

No reporte ningún fallo o recomiende nuevas ideas acerca de esta característica.
USAR BAJO SU PROPIO RIESGO.

Parámetros

apuntador_dll	El apuntador de una dll, retornado de una llamada previa a PluginOpen.
---------------	--

Valor de Retorno

Con Éxitos: Devuelve 1.
Al Fallar: Devuelve 0.

Comentarios

Ninguno.

Relativo

[PluginOpen](#)

Ejemplo

```
$handle = PluginOpen("ejemplo.dll")  
  
PluginFunc1(0.1, 0.2) ; llamará la función de plugin con 2 parámetros  
  
PluginClose($handle)
```

ProcessClose

Termina un proceso nombrado.

ProcessClose ("proceso")

Parámetros

proceso	El nombre o PID de el proceso a terminar.
---------	---

Valor de Retorno

- Con Éxito: Devuelve 1.
- Al Fallar: Devuelve 0 y establece @error a:
- 1 = OpenProcess falló
 - 2 = AdjustTokenPrivileges falló
 - 3 = TerminateProcess falló
 - 4 = No puede verificar si el proceso existe

Comentarios

Los nombres de procesos son ejecutables sin la ruta completa, ej., "notepad.exe" o "winword.exe"

Si procesos múltiples tienen el mismo nombre, el que tiene el más alto PID es el terminado--a pesar de como recientemente el proceso haya sido generado.

PID es el número único que identifica un Proceso. Un PID puede ser obtenido a través de los comandos ProcessExists o Run.

El proceso es consultado aproximadamente cada 250 milisegundos.

Relativo

[ProcessExists](#), [ProcessWait](#), [ProcessWaitClose](#), [Run](#), [WinClose](#), [ProcessList](#), [RunAs](#), [Shutdown](#), [WinKill](#)

Ejemplo

ProcessClose("notepad.exe")

*\$PID = ProcessExists("notepad.exe"); devolverá el PID o 0 si el proceso no es encontrado.
If \$PID Then ProcessClose(\$PID)*

ProcessExists

Revisa si un proceso especificado existe.

ProcessExists ("proceso")

Parámetros

proceso	El nombre o PID del proceso a revisar.
---------	--

Valor de Retorno

- Con Éxito: Devuelve el PID del proceso.
- Al Fallar: Devuelve 0 si proceso no existe.

Comentarios

Nombres de procesos son ejecutables sin la ruta completa, ej., "notepad.exe" o "winword.exe"

PID es el número único que identifica un Proceso.

El proceso es consultado aproximadamente cada 250 milisegundos.

Relativo

[ProcessClose](#), [ProcessWait](#), [ProcessWaitClose](#), [WinExists](#), [ProcessList](#)

Ejemplo

```
If ProcessExists("notepad.exe") Then  
    MsgBox(0, "Ejemplo", "Notepad está en ejecución.")  
EndIf
```

ProcessGetStats

Devuelve un arreglo sobre Memoria o información de IO(Entrada y Salida) de un proceso en ejecución.

ProcessGetStats (["proceso" [, tipo]])

Parámetros

proceso	[opcional] El nombre o PID del proceso a obtener información. Por defecto (-1) es el proceso actual.
tipo	[opcional] 0 = (por defecto) información de memoria, 1 = información de IO(E/S)

Valor de Retorno

Con Éxito: Un arreglo de información (Ver Comentarios)

Al Fallar: 0.

Comentarios

El arreglo devuelto es uni-dimensional y hace lo siguiente:

Tipo = 0 : Estadísticas de Memoria

\$arreglo[0] = WorkingSetSize
\$arreglo[1] = PeakWorkingSetSize

Tipo = 1 : Estadísticas de IO

\$arreglo[0] = número de operaciones de lectura realizadas.

\$arreglo[1] = número de operaciones de escritura realizadas.
\$arreglo[2] = número de operaciones de I/O(E/S) realizadas, que no son de operaciones de lectura y escritura.
\$arreglo[3] = número de bytes leídos.
\$arreglo[4] = número de bytes escritos.
\$arreglo[5] = número de bytes transferidos durante operaciones que no son de lectura y escritura.

Relativo

[ProcessList](#)

Ejemplo

; recupera la info de memoria de los procesos en ejecución
\$mem = ProcessGetStats()

; recupera la info de E/S de los procesos en ejecución
\$IO = ProcessGetStats(-1, 1)

ProcessSetPriority

Cambia la prioridad de un proceso

ProcessSetPriority ("proceso", prioridad)

Parámetros

proceso	El nombre o PID del proceso a revisar.
prioridad	Un bandera que determina que prioridad a establecer 0 - Inactivo/Bajo 1 - Por debajo de lo Normal 2 - Normal 3 - Por encima de lo Normal 4 - Alto 5 - Tiempo Real (Use con precaución, tal vez vuelva el sistema inestable)

Valor de Retorno

Con Éxito: Devuelve 1.
Al Fallar: Devuelve 0 y establece el valor de @error en 1. Puede establecer @error a 2 si se intenta usar una prioridad de clase no soportada.

Comentarios

Ninguno.

Relativo

[ProcessList](#), [_ProcessGetPriority](#)

Ejemplo

```
Run("Notepad.exe")
ProcessSetPriority("notepad.exe", 0)
; Notepad debería tener una prioridad Espera/Baja
```

ProcessList

Devuelve un arreglo listando los procesos actualmente en ejecución (nombres y PIDs)

ProcessList ([*"nombre"*])

Parámetros

nombre	[opcional] Si un nombre es dado solamente los procesos del mismo nombre serán devueltos.
--------	---

Valor de Retorno

Con Éxito: Un arreglo de nombres de procesos y sus PIDs (Ver Observaciones)

Al Fallar: @error es establecido a 1 cuando la lista no puede ser construida.

Comentarios

El arreglo devuelto es bi-dimensional y es compuesta de la siguiente manera:

\$arreglo[0][0] = Número de procesos

\$arreglo[1][0] = 1er. Nombre de proceso

\$arreglo[1][1] = 1er. ID de proceso (PID)

\$arreglo[2][0] = 2do. Nombre de proceso

\$arreglo[2][1] = 2do. ID de proceso (PID)

...

\$arreglo[n][0] = enésimo Nombre de proceso

\$arreglo[n][1] = enésimo ID de proceso (PID)

La lista puede estar vacía si \$arreglo[0][0] = 0. No se establece @error en este caso.

Relativo

[ProcessClose](#), [ProcessExists](#), [ProcessSetPriority](#), [ProcessWait](#), [ProcessWaitClose](#),
[ProcessGetStats](#), [WinGetProcess](#)

Ejemplo

```
; Lista todos los procesos
$list = ProcessList()
for $i = 1 to $list[0][0]
```

```

    msgbox(0, $list[$i][0], $list[$i][1])
next

; Lista sólo los procesos de notepad.exe
$list = ProcessList("notepad.exe")
for $i = 1 to $list[0][0]
    msgbox(0, $list[$i][0], $list[$i][1])
next

```

ProcessWait

Pausa el script ejecución hasta que proceso dado exista.

ProcessWait ("proceso" [, tiempoExpiración])

Parámetros

proceso	El nombre de el proceso a revisar.
tiempoExpiración	[opcional] Especifica cuánto tiempo a esperar (en segundos). Por defecto a esperar indefinidamente.

Valor de Retorno

Con Éxito: Devuelve el PID del proceso.

Al Fallar: Devuelve 0 si el tiempo de espera expiró.

Comentarios

Nombres de procesos son ejecutables sin la ruta completa, ej., "notepad.exe" o "winword.exe"

El proceso es consultado aproximadamente cada 250 milisegundos.

Esta función de procesos que no acepta un PID. Porque los PIDs son asignados al azar, esperar por cierto PID en particular que no existe, no tiene sentido.

Relativo

[ProcessClose](#), [ProcessExists](#), [ProcessWaitClose](#), [RunWait](#), [WinWait](#), [WinWaitActive](#), [ProcessList](#), [RunAsWait](#), [WinGetProcess](#)

Ejemplo

```
ProcessWait("notepad.exe")
```

ProcessWaitClose

Pausa el script en ejecución hasta que dado proceso termine.

ProcessWaitClose ("proceso" [, tiempoExpiración])

Parámetros

proceso	El nombre o PID del proceso a revisar.
tiempoExpiración	[opcional] Especifica cuánto tiempo a esperar (en segundos). Por defecto espera indefinidamente.

Valor de Retorno

Con Éxito: Devuelve 1 y establece @extended para el código de salida de los procesos.

Al Fallar: Devuelve 0 si el tiempo de espera expiró. En PID no válidos @error es establecido a no-cero y @extended es establecido a 0xFFFFFFFF.

Comentarios

Nombres de procesos son ejecutables sin la ruta completa, ej., "notepad.exe" o "winword.exe"

PID es el número único que identifica a un Proceso. Un PID puede ser obtenido a través de comandos como ProcessExists o Run.

El proceso es consultado aproximadamente cada 250 milisegundos.

El valor devuelto es 1 si el proceso no existe.

Relativo

[ProcessClose](#), [ProcessExists](#), [ProcessWait](#), [RunWait](#), [WinWaitClose](#), [ProcessList](#), [RunAsWait](#), [WinGetProcess](#)

Ejemplo

*;espera hasta que no exista una instancia de notepad.exe
ProcessWaitClose("notepad.exe")*

*; Esto esperará hasta que esta instancia en particular de notepad haya terminado
\$PID = Run("notepad.exe")
ProcessWaitClose(\$PID)*

Run

Ejecuta un programa externo.

Run ("programa" [, "directorio de trabajo" [, flag_mostrar[, opt_flag]]])

Parámetros

programa	La ruta completa del programa (EXE, BAT, COM, o PIF) a ejecutar (ver Comentarios).
directorio de Trabajo	[opcional] El directorio de trabajo actual. Este no es la ruta del programa.
flag_mostrar	[opcional] El bandera "show" del programa ejecutado: @SW_HIDE = Ventana oculta (o Palabra clave por defecto) @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada
opt_flag	[opcional] Controla varias opciones relativas a como los procesos padre y hijo interactúan. 0x1 (\$STDIN_CHILD) = Provee un apuntador al flujo o flujo del proceso hijo de STDIN 0x2 (\$STDOUT_CHILD) = Provee un apuntador al flujo o flujo de STDOUT 0x4 (\$STDERR_CHILD) = Provee un apuntador al flujo de STDERR 0x8 (\$STDERR_MERGED) = Provee el mismo apuntador para STDOUT y STDERR. Implica ambos \$STDOUT_CHILD y \$STDERR_CHILD. 0x10 (\$STDIO_INHERIT_PARENT) = Provee al hijo con el flujo STDIO del parent. Este flag no puede ser combinado con ningún otro flag de STDIO. Este flag es solo útil cuando el parent es compilado como una Aplicación de Consola. 0x10000 (\$RUN_CREATE_NEW_CONSOLE) = La consola del proceso hijo debería ser creado en una ventana propia en vez de utilizar la ventana parent. Este flag es solo útil cuando el parent es compilado como una Aplicación de Consola.

Valor de Retorno

Con Éxito: El PID del proceso que fue ejecutado.

Al Fallar: Devuelve 0 y establece el valor de @error en no-cero.

Comentarios

Las rutas con espacios necesitan ser encerradas entre comillas.

Para ejecutar comandos de DOS (consola), intente *Run(@ComSpec & "/c" y "nombre De Comando", "", @SW_HIDE);no se olvide de "" antes "/c"*

Después de ejecutar el programa solicitado el script continuará. Para detener la ejecución de secuencia de comandos hasta que el programa haya terminado utilice la función RunWait.

Proporcionar el Estándar de E/S(I/O) con el parámetro adecuado de valores permite la interacción con el proceso hijo a través de las funciones StderrRead, StdinWrite y StdoutRead. Combine el flag con valores (o use \$STDERR_CHILD, \$STDIN_CHILD & \$STDOUT_CHILD, definidos en Constants.au3) para gestionar más de una secuencia.

Con el fin de cerrar los flujos, las siguientes condiciones debe ser cumplidas: 1) El proceso hijo cerró su proceso al final del flujo (esto sucede cuando el proceso hijo se cierra). 2) AutoIt debe leer cualquier captura de flujos hasta que no hay más datos. 3) Si STDIN es provisto como

proceso hijo, StdinWrite() debe ser llamado cerrar el flujo. Una vez que todos los flujos son detectados como ya no es necesario, todos los recursos internos automáticamente serán liberados.

StdioClose puede ser usado para forzar a que el flujo de STDIO cierre.

Relativo

[RunWait](#), [RunAs](#), [RunAsWait](#), [ShellExecute](#), [ShellExecuteWait](#), [StderrRead](#), [StdinWrite](#), [StdoutRead](#), [StdioClose](#), [ProcessClose](#), [ConsoleRead](#)

Ejemplo

```
Run(@WindowsDir & "\Notepad.exe", "", @SW_MAXIMIZE)
```

RunWait

Ejecuta un programa externo y pausa la ejecución del script hasta que el programa finaliza.

RunWait ("programa" [, "directorio de trabajo" [, flag_mostrar [, opt_flag]]])

Parámetros

programa	La ruta completa del programa (EXE, BAT, COM, o PIF) a ejecutar (ver Comentarios).
directorio_de_trabajo	[opcional] El directorio de trabajo. Este no es la ruta del programa.
flag_mostrar	[opcional] El bandera "show" del programa ejecutado: @SW_HIDE = Ventana oculta (o Palabra clave por defecto) @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada
opt_flag	[optional] Controla varias opciones relativas a como los procesos padre e hijo interactúan. 0x10000 (\$RUN_CREATE_NEW_CONSOLE) = La consola del proceso hijo debería ser creado en una ventana propia en vez de utilizar la ventana padre. Este flag es solo útil cuando el padre es compilado como una Aplicación de Consola.

Valor de Retorno

Con Éxito: Devuelve el código de salida que del programa que fue ejecutado.

Al Fallar: Devuelve 0 y establece el valor de @error en no-cero.

Comentarios

Las rutas con espacios necesitan ser encerradas entre comillas.

Para ejecutar comandos de DOS (consola), intente `RunWait(@ComSpec & "/c" y "nombre de comando", "", @SW_HIDE)`; no se olvide de "" antes "/c"

Al ejecutar el programa solicitado el script se pausará. Para ejecutar un programa y continuar con el script utilice la función Run.

En algunos programas aparecerá que devuelve inmediatamente al script la ejecución, aunque todavía se ejecutando el programa; estos programas abren otros procesos - usted puede ser capaz de usar la función ProcessWaitClose al apuntador en estos casos.

Relativo

[ProcessWait](#), [ProcessWaitClose](#), [Run](#), [ShellExecute](#), [ShellExecuteWait](#), [RunAs](#), [RunAsWait](#)

Ejemplo

```
Local $val = RunWait(@WindowsDir & "\Notepad.exe", @WindowsDir, @SW_MAXIMIZE)
; script espera hasta que Notepad cierre
MsgBox(0, "Programa devuelve con el código de salida:", $val)
```

RunAs

Ejecuta un programa externo bajo el contexto de un usuario diferente.

RunAs ("NombreDeUsuario", "dominio", "contraseña", flag_acceso, "programa" [, "directorio de trabajo" [,flag_mostrar [, opt_flag]]]])

Parámetros

NombreDeUsuario	El nombre de usuario a conectar.
dominio	El dominio en el que será autenticado.
contraseña	La contraseña del usuario.
flag_acceso	0 - Sesión interactiva sin perfil. 1 - Sesión interactiva con perfil. 2 - Solamente credenciales de red. 4 - Hereda la llamada del entorno del proceso en lugar del usuario.
programa	La ruta completa del programa (EXE, BAT, COM, o PIF) a ejecutar (ver Comentarios).
directorio de Trabajo	[opcional] El directorio de trabajo actual. Este no es la ruta del programa.
flag_mostrar	[opcional] el flag "show" del programa ejecutado: @SW_HIDE = Ventana oculta (o Palabra clave por defecto) @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada
opt_flag	[opcional] Controla varias opciones relativas a como los procesos padre he

	<p>hijo interactúan</p> <p>0x1 (\$STDIN_CHILD) = Provee un apuntador al flujo del proceso hijo de STDIN</p> <p>0x2 (\$STDOUT_CHILD) = Provee un apuntador al flujo de STDOUT</p> <p>0x4 (\$STDERR_CHILD) = Provee un apuntador al flujo de STDERR</p> <p>0x8 (\$STDERR_MERGED) = Provee el mismo apuntador para STDOUT y STDERR. Implica ambos \$STDOUT_CHILD y \$STDERR_CHILD.</p> <p>0x10 (\$STDIO_INHERIT_PARENT) = Provee al hijo con el flujo STDIO del parent. Este flag no puede ser combinado con ningún otro flag de STDIO. Este flag es solo útil cuando el parent es compilado como una Aplicación de Consola.</p> <p>0x10000 (\$RUN_CREATE_NEW_CONSOLE) = La consola del proceso hijo debería ser creado en una ventana propia en vez de utilizar la ventana parent. Este flag es solo útil cuando el parent es compilado como una Aplicación de Consola.</p>
--	---

Valor de Retorno

Con Éxito: El PID del proceso que fue ejecutado.

Al Fallar: Devuelve 0 y establece el valor de @error en no-cero.

Comentarios

Las rutas con espacios necesitan ser encerradas entre comillas.

Es importante especificar un directorio de trabajo del usuario que usted está ejecutando como el que accede, de otro modo la función fallará.

Es recomendado que usted solamente cargue el perfil de usuario si está seguro que lo necesita. Hay una pequeña oportunidad de que un perfil puede ser pegado en la memoria bajo las condiciones adecuadas. Si un script usando RunAs() pasa para ser ejecutando con la cuenta de SYSTEM (por ejemplo, si el script está ejecutándose como un servicio) y el perfil de usuario es cargado, entonces usted debe tener cuidado de que el script se quede ejecutando hasta que el proceso hijo cierre.

Cuando se ejecuta como un administrador, el servicio de sesión secundaria (RunAs) debe ser activado sino la función fallará. Este no aplica cuando se está ejecutando como la cuenta de SYSTEM.

Después de ejecutar el programa solicitado el script continuará. Para detener la ejecución de secuencia de comandos hasta que el programa haya terminado utilice la función RunWait.

Proporcionar el Estándar de E/S(I/O) con el parámetro adecuado de valores permite la interacción con el proceso hijo a través de las funciones StderrRead, StdinWrite y StdoutRead. Combine el flag con valores (o use \$STDERR_CHILD, \$STDIN_CHILD & \$STDOUT_CHILD, definidos en Constants.au3) para gestionar más de una secuencia.

Con el fin de que cerrar flujos, las siguientes condiciones debe ser cumplidas: 1) El proceso hijo cerró su proceso al final del flujo (esto sucede cuando el proceso hijo se cierra). 2) Autolt debe leer cualquier captura de flujos hasta que no hay más datos. 3) Si STDIN es provisto como proceso hijo, StdinWrite() debe ser llamado cerrar el flujo. Una vez que todos los flujos son

detectados como ya no es necesario, todos los recursos internos automáticamente serán liberados.

StdioClose puede ser usado para forzar a que el flujo de STDIO cierre.

Opciones de "cargar perfil" y "solamente credenciales de red" son incompatibles. Usando ambas producirá resultados impredecibles.

Existe un asunto en la generación de Windows XP generation de Windows que impide la redirección del STDIO redirection y el flag_mostrar en el directorio de trabajo. Ver el artículo de la Base de Conocimientos de Microsoft KB818858 para más información acerca de cuales versiones son afectadas así como también un correción para este asunto. Los usuarios con Windows 2000, Windows XP SP2 o después, o Windows Vista no son afectados.

Relativo

[Run](#), [RunWait](#), [RunAsWait](#), [ShellExecute](#), [ShellExecuteWait](#), [StderrRead](#), [StdinWrite](#), [StdoutRead](#), [StdioClose](#), [ProcessClose](#)

Ejemplo

; Llena el nombre de usuario y contraseña apropiados para su sistema.

Local \$sUserName = "NombreDeUsuario"

Local \$sPassword = "Password"

; Ejecuta un comando como el otro usuario.

RunAs(\$sUserName, @ComputerName, \$sPassword, 0, @ComSpec, @SystemDir)

RunAsWait

Ejecuta un programa externo bajo el contexto de un usuario diferente y pausa la ejecución del script hasta que el programa finaliza.

RunAsWait ("NombreDeUsuario", "dominio", "contraseña", flag_acceso, "programa" [, "directorio de trabajo" [,flag_mostrar [, opt_flag]]])

Parámetros

NombreDeUsuario	El nombre de usuario a conectar.
dominio	El dominio en el que será autenticado.
contraseña	La contraseña del usuario.
flag_acceso	0 - Sesión interactiva sin perfil. 1 - Sesión interactiva con perfil. 2 - Solamente credenciales de red. 4 - Hereda la llamada del entorno del proceso en lugar del usuario.
programa	La ruta completa del programa (EXE, BAT, COM, o PIF) a ejecutar (ver Comentarios).

directorio_de_Trabajo	[opcional] El directorio de trabajo actual. Este no es la ruta del programa.
flag_mostrar	[opcional] El bandera "show" del programa ejecutado: @SW_HIDE = Ventana oculta (o Palabra clave por defecto) @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada
opt_flag	[opcional] Controla varias opciones relativas a como los procesos padre he hijo interactúan. 0x10000 (\$RUN_CREATE_NEW_CONSOLE) = La consola del proceso hijo debería ser creado en una ventana propia en vez de utilizar la ventana padre. Este flag es solo útil cuando el padre es compilado como una Aplicación de Consola.

Valor de Retorno

Con Éxito: Devuelve el código de salida que del programa que fue ejecutado.

Al Fallar: Devuelve 0 y establece el valor de @error en no-cero.

Comentarios

Las rutas con espacios necesitan ser encerradas entre comillas.

Es importante especificar un directorio de trabajo del usuario que usted está ejecutando como el que accede, de otro modo la función fallará.

Es recomendado que usted solamente cargue el perfil de usuario si está seguro que lo necesita. Hay una pequeña oportunidad de que un perfil puede ser pegado en la memoria bajo las condiciones adecuadas. Si un script usando RunAs() pasa para ser ejecutado con la cuenta de SYSTEM (por ejemplo, si el script está ejecutándose como un servicio) y el perfil de usuario es cargado, entonces usted debe tener cuidado de que el script se quede ejecutando hasta que el proceso hijo cierre.

Cuando se ejecuta como un administrador, el servicio de sesión secundaria (RunAs) debe ser activado sino la función fallará. Este no aplica cuando se está ejecutando como la cuenta de SYSTEM.

Al ejecutar el programa solicitado el script se pausará. Para ejecutar un programa y continuar con el script utilice la función RunAs.

En algunos programas aparecerá que devuelve inmediatamente al script la ejecución, aunque todavía se ejecutando el programa; estos programas abren otros procesos - usted puede ser capaz de usar la función ProcessWaitClose al apuntador en estos casos.

Opciones de "cargar perfil" y "solamente credenciales de red" son incompatibles. Usando ambas producirá resultados impredecibles.

Relativo

[ProcessWait](#), [ProcessWaitClose](#), [Run](#), [RunWait](#), [ShellExecute](#), [ShellExecuteWait](#), [RunAs](#)

Ejemplo

```
; Llena el nombre de usuario y contraseña apropiados para su sistema.  
Local $sUserName = "NombreDeUsuario"  
Local $sPassword = "Password"  
  
; Ejecuta un comando como el otro usuario.  
RunAs($sUserName, @ComputerName, $sPassword, 0, @ComSpec, @SystemDir)  
  
; Espera a que el proceso cierre.  
ProcessWaitClose($pid)  
  
; Muestra un mensaje.  
MsgBox(0, "", "El proceso que se esperaba cerrarse ha terminado.")
```

ShellExecute

Ejecuta un programa externo usando la API de ShellExecute.

ShellExecute ("nombre de archivo" [, "Parámetros" [, "directorio de trabajo" [, "verbo" [, flagDeAparicion]]]])

Parámetros

nombre de archivo	El nombre del archivo a ejecutar (EXE, .txt, .lnk, etc).
Parámetros	[opcional] Opción para enviar parámetros al programa ejecutado. No usará ninguno si se deja en blanco ("")
Directorio_de_trabajo	[opcional] El directorio de trabajo. Dejando en vacío ("") usa el directorio actual de trabajo.
verbo	[opcional] El "verbo" a utilizar, verbos comunes incluyen: open = Abre el archivo especificado. El archivo puede ser un archivo ejecutable, un documento, o una carpeta edit = Abre el editor y abre el documento para edición. Si "nombre de archivo" no es un documento, la función fallará print = Imprime el documento especificado. Si "nombre de archivo" no es un documento, la función fallará properties = Muestra el archivo o propiedad de carpeta
flagDeAparición	Ver Comentarios para más información del comportamiento por defecto en caso de no ser especificado el verbo. [opcional] La bandera de aparición o "show" define como se verá la ventana del programa ejecutado: @SW_HIDE = Ventana oculta @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada

Valor de Retorno

- Con Éxito: Devuelve 1.
Al Fallar: Devuelve 0 y establece el valor de @error a no-cero.

Comentarios

Después de ejecutar el programa solicitado el script continúa. Para pausar la ejecución del script hasta que el programa generado finalice usar la función ShellExecuteWait en su lugar.

Cuando el verbo no es especificado el utilizado es el por defecto. El verbo por defecto es el configurado en el registro. Si en el registro no existe ningún verbo establecido por defecto entonces el verbo "open" verb es usado. Si el verbo "open" no esta presente entonces el primer verbo listado en el registro es usado(excepto en Windows 2000).

Relativo

[ShellExecuteWait](#), [Run](#), [RunWait](#), [RunAs](#), [RunAsWait](#)

Ejemplo

```
; Abre Notepad  
ShellExecute("Notepad.exe")
```

```
; Abre un archivo de texto (.txt) con su editor predeterminado  
ShellExecute("myfile.txt", "", "@ScriptDir, "edit")
```

ShellExecuteWait

Ejecuta un programa externo usando la API de ShellExecute y pausa la ejecución del script hasta que finalice.

ShellExecuteWait ("nombre_de_archivo" [, "Parámetros" [, "directorio_de_trabajo" [, "verbo" [, flagDeAparición]]]])

Parámetros

nombre_de_archivo	El nombre del archivo a ejecutar (EXE, .txt, .lnk, etc).
Parámetros	[opcional] Opción para enviar parámetros al programa ejecutado. No usará ninguno si se deja en blanco ("")
Directorio_de_Trabajo	[opcional] El directorio de trabajo. Dejando en vacío ("") usa el directorio actual de trabajo.
verbo	[opcional] El "verbo" a utilizar, verbos comunes incluyen: open = Abre el archivo especificado. El archivo puede ser un archivo ejecutable, un documento, o una carpeta edit = Abre el editor y abre el documento para edición. Si "nombre de archivo" no es un documento, la función fallará print = Imprime el documento especificado. Si "nombre de archivo" no es un documento, la función fallará

	properties = Muestra el archivo o propiedad de carpeta Ver Comentarios para más información del comportamiento por defecto en caso de no ser especificado el verbo.
flagDeAparición	[opcional] La bandera de aparición o "show" define como se verá la ventana del programa ejecutado: @SW_HIDE = Ventana oculta @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada

Valor de Retorno

Con Éxito: Devuelve el código de salida del programa que fué ejecutado.
Al Fallar: Devuelve 0 y establece el valor de @error a no-cero.

Comentarios

Después de ejecutar el programa solicitado el script pausa hasta que el programa solicitado termina.

Cuando el verbo no es especificado el utilizado es el por defecto. El verbo por defecto es el configurado en el registro. Si en el registro no existe ningún verbo establecido por defecto entonces el verbo "open" verb es usado. Si el verbo "open" no esta presente entonces el primer verbo listado en el registro es usado(excepto en Windows 2000).

Relativo

[ShellExecute](#), [Run](#), [RunWait](#), [RunAs](#), [RunAsWait](#)

Ejemplo

```
$val = ShellExecuteWait("Notepad.exe")
; el script espera que Notepad cierra
MsgBox(0, "Programa devuelto con el código de salida:", $val)
```

Shutdown

Apaga el sistema.

Shutdown (código [, razón])

Parámetros

código	Una combinación de códigos de apagado. Ver Comentarios.
razón	[opcional] Código de razón de terminación de usuario.

Valor de Retorno

- Con Éxito: Devuelve 1.
 Al Fallar: Devuelve 0 y @error es establecido a GetLastError().

Comentarios

El código de apagado es una combinación de los siguientes valores:
 0 = Cerrar sesión (Logoff)
 1 = Cierra el sistema (Shutdown)
 2 = Reiniciar (Reboot)
 4 = Forzar (Force)
 8 = Apaga el sistema
 16= Forzar si se cuelga
 32= Suspender (Standby)
 64= Hibernación

Agregar los valores requeridos juntos. Para cerrar y apagar, por ejemplo, el código debería ser 9 (shutdown + power down = 1 + 8 = 9).

Suspender o Hibernar son ignorados si otros códigos son establecidos.

Relativo

[ProcessClose](#)

Ejemplo

Shutdown(6) ;Fuerza un reinicio

StderrRead

Lee del flujo de STDERR de un proceso hijo previamente ejecutado.

StderrRead (id_del_proceso[, revisión = false[, binario = false]])

Parámetros

id_del_proceso	El ID del proceso de un proceso hijo, como es devuelto por una llamada previa de Run.
revisión	[opcional] Si el valor es <i>true</i> , es decir, verdadero la función no remueve los caracteres leídos del flujo.
binario	[opcional] Si el valor <i>true</i> , es decir, verdadero la función lee la información como binario en vez de texto (por defecto es texto).

Valor de Retorno

Con Éxito: Devuelve la información leída. @extended contiene el número de bytes leídos.

Al Fallar: Establece @error a no-cero si EOF es alcanzado, STDERR no fue redirigido por el proceso u otro error.

Comentarios

StderrRead lee el flujo de la consola estándar de un proceso hijo, que normalmente es usado por aplicaciones de consola para escribir a la pantalla. Durante la llamada a Run para el proceso hijo usted desea leer, el parámetro STD I/O debe tener incluido el valor de \$STDERR_CHILD (4) para que esta función trabaje apropiadamente (vea la función [Run](#)).

StderrRead no bloquea, lo devolverá inmediatamente. Con el fin de obtener toda la información, debe ser llamado en un ciclo.

Revisando sobre el flujo no elimina los datos de la memoria, sin embargo, no devolverá la información disponible como es normalmente.

Por defecto, los datos o información es devuelta en formato de texto. Usando la opción binaria, los datos serán devueltos en formato binario.

Relativo

[StdoutRead](#), [StdinWrite](#), [StdioClose](#), [Run](#), [RunAs](#)

Ejemplo

```
; Demuestra StdoutRead()
#include <Constants.au3>

Local $foo = Run(@ComSpec & " /c dir foo.bar", @SystemDir, @SW_HIDE, $STDERR_CHILD + $STDOUT_CHILD)
Local $line
While 1
    $line = StdoutRead($foo)
    If @error Then ExitLoop
    MsgBox(0, "STDOUT lee:", $line)
Wend

While 1
    $line = StderrRead($foo)
    If @error Then ExitLoop
    MsgBox(0, "STDERR lee:", $line)
Wend

MsgBox(0, "Debug", "Saliendo...")
```

StdinWrite

Escribe un número de caracteres al flujo de STDIN de un proceso hijo previamente ejecutado.

StdinWrite (id_del_proceso[, datos])

Parámetros

id_del_proceso	El ID del proceso de un proceso hijo, devuelto por una llamada previa a Run.
datos	[opcional] Los datos que usted desea escribir. Este puede ser como texto o binario.

Valor de Retorno

Con Éxito: Devuelve el número de caracteres escritos.

Al Fallar: Establece @error a no-cero si STDIN no fue redirigido por el proceso u otro error.

Comentarios

StderrRead escribe el flujo de la consola estándar de un proceso hijo, que normalmente es usado por aplicaciones de consola para escribir a la pantalla. Durante la llamada a Run para el proceso hijo usted desea leer, el parámetro STD I/O debe tener incluido el valor de \$STDERR_CHILD (4) para que esta función trabaje apropiadamente (vea la función [Run](#)).

El segundo parámetro opcional es la cadena que usted desea StdinWrite escribir al flujo. Si la función es llamada sin un segundo argumento, StdinWrite cierra el flujo y lo invalida para escritura posterior.

El flujo es de principio a fin en memoria con un tamaño arbitrario limitado; si en cualquier tiempo cuando esta función es llamada (a menos sea llamada para cerrar el flujo) no hay más espacio para más caracteres para que sean escritos al flujo, la función StdinWrite función bloqueará (pausa) y no vuelve hasta que el proceso hijo cierra el flujo o lee lo suficientes caracteres del flujo para permitir que el procedimiento de escritura complete. Esto significa que el proceso de Autolt será detenido, y no procesará hotkeys, mensajes GUI, etc. hasta que el proceso hijo lea del flujo de STDIN.

Los caracteres son convertidos a ANSI antes de ser escritos.

Los datos binarios son escritos tal como son. Estos no son convertidos a cadena. Para imprimir una representación hexadecimal de un dato binario, use la función String() para una conversión explícita a cadena.

Relativo

[StdoutRead](#), [StderrRead](#), [StdioClose](#), [Run](#), [RunAs](#)

Ejemplo

; Demuestra el uso de StdinWrite()

#include <Constants.au3>

```
Local $foo = Run("sort.exe", @SystemDir, @SW_HIDE, $STDIN_CHILD + $STDOUT_CHILD)
; Escribe una cadena para ser ordenada al proceso hijo de sort.exe STDIN
StdinWrite($foo, "rat" & @CRLF & "cat" & @CRLF & "bat" & @CRLF)
```

```

; Llamando sin el segundo argumento cierra el flujo
StdinWrite($foo)

; Lee del proceso hijo STDOUT y lo muestra
Local $data
While True
    $data &= StdoutRead($foo)
    If @error Then ExitLoop
    Sleep(25)
WEnd
MsgBox(0, "Debug", $data)

```

StdioClose

Cierra todos los recursos asociados con un proceso previamente ejecutado redirección de STDIO.

StdioClose (id_del_proceso)

Parámetros

id_del_proceso	El ID del proceso de un proceso hijo, devuelto por una llamada previa a Run.
----------------	--

Valor de Retorno

Con Éxito: No-cero.

Al Fallar: 0 si el proceso no tiene redirección STDIO o ya fue cerrado.

Comentarios

Este función cierra todos los apuntadores y las emisiones relacionadas con todos los recursos de STDIO. No sería posible leer datos del proceso STDIO. Datos pendientes se perderán.

Relativo

[StdoutRead](#), [StderrRead](#), [StdinWrite](#), [Run](#), [RunAs](#)

Ejemplo

```

; Demuestra StdioClose()
#include <Constants.au3>

Local $pid = Run(@ComSpec & "/c dir foo.bar", @SystemDir, @SW_HIDE, $STDERR_MERGED +
$STDOUT_CHILD)
StdioClose($pid)

; Sin datos a ser leídos porque se han cerrados todos los flujos.
Local $line
While 1

```

```

$line = StdoutRead($pid)
If @error Then ExitLoop
MsgBox(0, "STDOUT lee:", $line)
Wend

While 1
$line = StderrRead($pid)
If @error Then ExitLoop
MsgBox(0, "STDERR lee:", $line)
Wend

MsgBox(0, "Debug", "Saliendo...")

```

StdoutRead

Lee del flujo STDOUT de un proceso hijo previamente ejecutado.

StdoutRead (id_del_proceso[, revisión = false[, binario = false]])

Parámetros

id_del_proceso	El ID del proceso hijo, devuelto por una llamada previa a Run.
revisión	[opcional] Si el valor es <i>true</i> , es decir, verdadero la función no remueve los caracteres leídos del flujo.
binario	[opcional] Si el valor <i>true</i> , es decir, verdadero la función lee la información como binario en vez de texto (por defecto es texto)

Valor de Retorno

- Con Éxito: Devuelve la información leída. @extended contiene el número de bytes leídos.
 Al Fallar: Establece @error a no-cero si EOF es alcanzado, STDOUT no fue redirigido por el proceso u otro error.

Comentarios

StdoutRead lee el flujo de la consola estándar de un proceso hijo, que normalmente es usado por aplicaciones de consola para escribir a la pantalla. Durante la llamada a Run para el proceso hijo usted desea leer, el parámetro STD I/O parámetro debe tener incluido el valor de \$STDOUT_CHILD (2) para que esta función trabaje apropiadamente (vea la función [Run](#)).

StdoutRead no bloquea, lo devolverá inmediatamente. Con el fin de obtener toda la información, debe ser llamado en un ciclo.

Revisando sobre el flujo no elimina los datos de la memoria, sin embargo, no devolverá la información disponible como es normalmente.

Por defecto, los datos o información es devuelta en formato de texto. Usando la opción binaria, los datos serán devueltos en formato binario.

Relativo

[StderrRead](#), [StdinWrite](#), [StdioClose](#), [Run](#), [RunAs](#)

Ejemplo

```
; Demuestra StdoutRead()
#include <Constants.au3>

Local $foo = Run(@ComSpec & "/c dir foo.bar", @SystemDir, @SW_HIDE, $STDERR_CHILD + $STDOUT_CHILD)
Local $line
While 1
    $line = StdoutRead($foo)
    If @error Then ExitLoop
    MsgBox(0, "STDOUT lee:", $line)
Wend

While 1
    $line = StderrRead($foo)
    If @error Then ExitLoop
    MsgBox(0, "STDERR lee:", $line)
Wend

MsgBox(0, "Debug", "Saliendo...")
```

Referencia de Administración de Registro

A continuación se muestra una lista completa de las funciones de Administración de Registro disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
RegDelete	Elimina una clave del registro.
RegEnumKey	Lee el nombre de una subclave de acuerdo a su instancia.
RegEnumVal	Lee el nombre de un valor de acuerdo a su instancia.
RegRead	Lee un valor del registro.
RegWrite	Crea una clave o valor en el registro.

RegDelete

Elimina una clave del registro.

RegDelete ("nombredelclave" [, "nombredelvalor"])

Parámetros

nombredelclave	La clave de registro a borrar.
nombredelvalor	[opcional] El nombre del valor a borrar.

Valor de Retorno

Con Éxito: Devuelve 1.

Especial: Devuelve 0 si el clave/valor no existe.

Al Fallar: Devuelve 2 si hubo un error al eliminar clave/valor.

@error puede ser establecerse a los siguientes valores :

1 si está deshabilitado abrir la clave solicitada

2 si está deshabilitado abrir la clave principal solicitada

3 si está deshabilitado conectar remotamente al registro

-1 si está deshabilitado borrar el valor solicitado

-2 si está deshabilitado borrar clave/valor solicitado

Comentarios

Una clave del Registro debe comenzar con "HKEY_LOCAL_MACHINE" ("HKLM") o "HKEY_USERS" ("HKU") o "HKEY_CURRENT_USER" ("HKCU") o "HKEY_CLASSES_ROOT" ("HKCR") o "HKEY_CURRENT_CONFIG" ("HKCC") .

Cuando se ejecuta en 64-bit de Windows si desea eliminar una clave o valor específico para el entorno de 64 bits tiene que el sufijo de HK... con 64 es decir, HKLM64.

Para acceder al valor (**por defecto**) use "" (una cadena en blanco) para 'nombredelvalor'.

Eliminar del registro es potencialmente peligroso--por favor, tener cuidado!

Es posible acceder a los registros a distancia mediante el uso de un nombredelclave en la forma "\computadora\nombredelclave". Para utilizar esta característica debe tener los derechos de acceso correctos.

Relativo

[RegRead](#), [RegWrite](#), [RegEnumKey](#), [RegEnumVal](#)

Ejemplo

```
RegDelete("HKEY_CURRENT_USER\Software\Test", "TestKey")
```

```
RegWrite("HKEY_CURRENT_USER\Software\Test")
RegWrite("HKEY_CURRENT_USER\Software\Test1")
```

RegEnumKey

Lee el nombre de una subclave de acuerdo a su instancia.

RegEnumKey ("nombredelclave", instancia)

Parámetros

nombredelclave	El clave del registro a leer.
instancia	La instancia de clave base-1 a recuperar.

Valor de Retorno

- Con Éxito: Devuelve el nombre de la subclave solicitada.
Al Fallar: Devuelve una cadena vacía "" y establece @error a:
1 si está deshabilitado abrir la clave solicitada
2 si está deshabilitado abrir la clave principal solicitada
3 si está deshabilitado conectar remotamente al registro
-1 si está deshabilitado recuperar la subclave solicitada (instancia de clave fuera de rango)

Comentarios

Una clave del Registro debe comenzar con "HKEY_LOCAL_MACHINE" ("HKLM") o "HKEY_USERS" ("HKU") o "HKEY_CURRENT_USER" ("HKCU") o "HKEY_CLASSES_ROOT" ("HKCR") o "HKEY_CURRENT_CONFIG" ("HKCC").

Cuando se está ejecutando en Windows 64-bit si usted desea enumerar una clave específica al entorno de 64 bit usted debe tener el sufijo en HK... con 64 ej. HKLM64.

Relativo

[RegEnumVal](#), [RegDelete](#), [RegWrite](#)

Ejemplo

```
For $i= 1 to 10
  $var = RegEnumKey("HKEY_LOCAL_MACHINE\SOFTWARE", $i)
  If @error <> 0 then ExitLoop
  MsgBox(4096, "SubClave #" & $i & " bajo HKLM\Software: ", $var)
Next
```

RegEnumVal

Lee el nombre de un valor de acuerdo a su instancia.

RegEnumVal ("nombredeclave", instancia)

Parámetros

nombredeclave	El clave del registro a leer.
instancia	La instancia de clave base-1 a recuperar.

Valor de Retorno

Con Éxito: Devuelve el valor del registro solicitado. @EXTENDED es establecido al tipo de valor.

Al Fallar: Devuelve una cadena vacía "" y establece @error a:
1 si está deshabilitado abrir la clave solicitada
2 si está deshabilitado abrir la clave principal solicitada
3 si está deshabilitado conectar remotamente al registro
-1 si está deshabilitado recuperar la subclave solicitada (instancia de clave fuera de rango)

Comentarios

Una clave del Registro debe comenzar con "HKEY_LOCAL_MACHINE" ("HKLM") o "HKEY_USERS" ("HKU") o "HKEY_CURRENT_USER" ("HKCU") o "HKEY_CLASSES_ROOT" ("HKCR") o "HKEY_CURRENT_CONFIG" ("HKCC").

Cuando se está ejecutando en Windows 64-bit si usted desea enumerar una clave específica al entorno de 64 bit usted debe tener el sufijo en HK... con 64 ej. HKLM64.

[RegEnumKey](#), [RegDelete](#), [RegWrite](#)

Ejemplo

```
For $i = 1 to 100
$var = RegEnumVal("HKEY_LOCAL_MACHINE\SOFTWARE\AutoIt v3\Autoit", $i)
if @error <> 0 Then ExitLoop
MsgBox(4096, "Nombre de valor #" & $i & " bajo clave de AutoIt3", $var)
next
```

RegRead

Lee un valor del registro.

RegRead ("NombreDeClave", "NombreDeValor")

Parámetros

nombredeclave	El clave de registro a leer.
NombreDeValor	El valor a leer.

Valor de Retorno

Con Éxito: Devuelve el valor solicitado del registro. @EXTENDED es establecido al tipo del valor \$REG_... . Estos tipos son definidos en el archivo incluido de "Constants.au3".

Al Fallar: Devuelve "" y establece la bandera @error a:
1 si está deshabilitado abrir la clave solicitada
2 si está deshabilitado abrir la clave principal solicitada
3 si está deshabilitado conectarse remotamente al registro
-1 si está deshabilitado abrir el valor solicitado
-2 si tipo de valor no está soportado

Comentarios

Una clave del Registro debe comenzar con "HKEY_LOCAL_MACHINE" ("HKLM") o "HKEY_USERS" ("HKU") o "HKEY_CURRENT_USER" ("HKCU") o "HKEY_CLASSES_ROOT" ("HKCR") o "HKEY_CURRENT_CONFIG" ("HKCC").

Cuando se está ejecutando en Windows 64-bit si usted desea enumerar una clave específica al entorno de 64 bit usted debe tener el sufijo en HK... con 64 ej. HKLM64.

AutoIt soporta claves de registro del tipo REG_BINARY, REG_SZ, REG_MULTI_SZ, REG_EXPAND_SZ, y REG_DWORD.

Para acceder al valor (**por defecto**) use "" (una cadena en blanco) para nombredelvalor.

Cuando se lee una clave REG_BINARY, el resultado es un tipo de dato binario (en versiones previas este era una cadena de valores hexadecimales).

Cuando se lee una clave REG_MULTI_SZ, devolverá múltiples entradas separados por @LF - usar StringSplit(..., @LF) para obtener cada entrada.

Es posible acceder a los registros a distancia mediante el uso de un 'nombredelclave' en la forma "\\computadora\\nombredelclave". Para utilizar esta característica debe tener los derechos de acceso correctos.

Relativo

[RegDelete](#), [RegWrite](#), [StringSplit](#)

Ejemplo

```
$var = RegRead("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion",
"ProgramFilesDir")
MsgBox(4096, "Archivos de programa están en:", $var)
```

RegWrite

Crea una clave o valor en el registro.

RegWrite ("NombreDeClave" [, "NombreDeValor" [, "tipo" [, valor]]])

Parámetros

NombreDeClave	La clave del registro a escribir. Si no se especifican parámetros esta clave simplemente será creada.
NombreDeValor	[opcional] El nombre del valor a escribir.
tipo	[opcional] Tipo de clave a escribir: "REG_SZ", "REG_MULTI_SZ", "REG_EXPAND_SZ", "REG_DWORD", "REG_QWORD", o "REG_BINARY".
valor	[opcional] El valor a escribir.

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si hubo un error escribiendo la clave o valor del registro.
@error puede ser establecido a los siguientes valores:
1 si está deshabilitado abrir la clave solicitada
2 si está deshabilitado abrir la clave principal solicitada
3 si está deshabilitado conectarse remotamente al registro
-1 si está deshabilitado abrir el valor solicitado
-2 si tipo de valor no está soportado

Comentarios

Una clave del Registro debe comenzar con "HKEY_LOCAL_MACHINE" ("HKLM") o "HKEY_USERS" ("HKU") o "HKEY_CURRENT_USER" ("HKCU") o "HKEY_CLASSES_ROOT" ("HKCR") o "HKEY_CURRENT_CONFIG" ("HKCC").

Cuando se está ejecutando en Windows 64-bit si usted desea enumerar una clave específica al entorno de 64 bit usted debe tener el sufijo en HK... con 64 ej. HKLM64.

Autolt soporta claves de registro del tipo REG_BINARY, REG_SZ, REG_MULTI_SZ, REG_EXPAND_SZ, y REG_DWORD.

Para acceder el (**Default**) valor use "" (una cadena en blanco) para el NombreDeValor.

Para acceder al valor (**por defecto**) use "" (una cadena en blanco) para NombreDeValor.

Cuando se escribe una clave REG_MULTI_SZ con múltiples entradas, debe separarse con @LF. El valor no debe terminar con un @LF a menos que usted quiere dejar una cadena vacía al final (ver ejemplo).

Es posible acceder a los registros a distancia mediante el uso de un NombreDeClave en la forma "\\\computadora\nombredeclave". Para utilizar esta característica debe tener los derechos de acceso correctos en NT/2000/XP/2003.

Relativo

[RegDelete](#), [RegRead](#), [RegEnumKey](#), [RegEnumVal](#)

Ejemplo

```
; Escribe un valor simple de REG_SZ
RegWrite("HKEY_CURRENT_USER\Software\Test", "TestKey", "REG_SZ", "Hola esto es una prueba")

; Escribe valor REG_MULTI_SZ de "line1" y "line2"
RegWrite("HKEY_CURRENT_USER\Software\Test", "TestKey1", "REG_MULTI_SZ", "line1" & @LF & "line2")

; Escribe el valor REG_MULTI_SZ de "line1"
RegWrite("HKEY_CURRENT_USER\Software\Test", "TestKey2", "REG_MULTI_SZ", "line1")

; siempre añade una cadena vacía nula
```

```
RegWrite("HKEY_CURRENT_USER\Software\Test", "TestKey3", "REG_MULTI_SZ", "line1" &
@LF & "line2" & @LF)
RegWrite("HKEY_CURRENT_USER\Software\Test", "TestKey4", "REG_MULTI_SZ", "line1" &
@LF & @LF & "line2" & @LF)

; vacía REG_MULTI_SZ
RegWrite("HKEY_CURRENT_USER\Software\Test", "TestKey5", "REG_MULTI_SZ", "")

; crea solo la clave
RegWrite("HKEY_CURRENT_USER\Software\Test1")
```

Referencia de Administración de Cadena

A continuación se muestra una lista completa de las funciones de Administración de Cadena disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
StringAddCR	Toma una cadena de texto y todos los caracteres de línea prefijos (Chr(10)) con un carácter de retorno de carro (Chr(13))
StringCompare	Compara dos cadenas de texto con opciones.
StringInStr	Revisa si una cadena contiene una subcadena dada.
StringIsAlNum	Revisa si una cadena contiene solamente caracteres alfanuméricos.
StringIsAlpha	Revisa si una cadena contiene solamente caracteres alfabéticos.
StringIsASCII	Revisa si una cadena contiene solamente caracteres ASCII en el rango 0x00 - 0x7f (0 - 127).
StringIsDigit	Revisa si una cadena contiene solamente caracteres de dígitos (0-9).
StringIsFloat	Revisa si una cadena es un número tipo punto flotante.
StringFormat	Devuelve una cadena de texto con formato (similar a la función sprintf() de C)
StringFromASCIIArray	Convierte un arreglo de códigos ASCII a una cadena.
StringIsInt	Revisa si una cadena es un entero.
StringIsLower	Revisa si una cadena contiene solamente letras minúsculas.
StringIsSpace	Revisa si una cadena contiene solamente caracteres de espacios en blanco.

<u>StringIsUpper</u>	Revisa si una cadena contiene solamente letras mayúsculas.
<u>StringIsXDigit</u>	Revisa si una cadena contiene solamente dígitos (0-9) y caracteres hexadecimales (A-F)
<u>StringLeft</u>	Devuelve un número de caracteres a partir del lado izquierdo de una cadena de texto.
<u>StringLen</u>	Devuelve el número de caracteres en una cadena de texto.
<u>StringLower</u>	Convierte a cadena a minúsculas.
<u>StringMid</u>	Extrae un número de caracteres de una cadena de texto.
<u>StringRegExp</u>	Revisa si una cadena encaja en un patrón dado de expresión regular.
<u>StringRegReplace</u>	Reemplaza texto en un cadena basado en un expresión regular.
<u>StringReplace</u>	Reemplaza subcadenas en una cadena.
<u>StringRight</u>	Devuelve un número de caracteres del lado izquierdo de una cadena.
<u>StringSplit</u>	Divide una cadena en varias partes de acuerdo a un delimitador dado.
<u>StringStripCR</u>	Remueve todos los valor de retorno de carro (Chr(13)) de una cadena.
<u>StringStripWS</u>	Recorta los espacios en blanco de una cadena.
<u>StringToASCIIArray</u>	Convierte una cadena a un arreglo conteniendo el código ASCII de cada carácter.
<u>StringTrimLeft</u>	Corta un número de caracteres del lado izquierdo de una cadena.
<u>StringTrimRight</u>	Corta un número de caracteres del lado derecho de una cadena.
<u>StringUpper</u>	Convierte una cadena a mayúsculas.

StringAddCR

Toma una cadena de texto y todos los caracteres de línea prefijos (Chr(10)) con un carácter de retorno de carro (Chr(13)).

StringAddCR ("cadena de texto")

Parámetros

cadena de texto	La cadena de texto a convertir.
-----------------	---------------------------------

Valor de Retorno

Devuelve la cadena de texto con todas las instancias de los caracteres de alimentación de línea (@LF) prefijados con un carácter de retorno de carro (@CR).

Comentarios

Ninguno.

Relativo

[StringStripCR](#), [StringReplace](#)

Ejemplo

```
$old = "Notepad" & @LF & "espera" & @LF & "un texto CRLF."  
$new = StringAddCR($old)
```

StringCompare

Compara dos cadenas de texto con opciones.

StringCompare ("cadena de texto1", "cadena de texto2" [, casoSensitivo])

Parámetros

cadena de texto1	La primer cadena a evaluar.
cadena de texto2	La segunda cadena a evaluar.

casoSensitivo	<p>[opcional] Bandera para indicar si las operaciones deberían ser caso sensitivo.</p> <p>0 = no caso sensitivo, usando el user's locale (por defecto)</p> <p>1 = caso sensitivo</p> <p>2 = no caso sensitivo, usando una comparación básica/rápida</p>
---------------	--

Valor de Retorno

- 0 cadena de texto1 y cadena de texto2 son iguales
- > 0 cadena de texto1 es mayor que cadena de texto2
- < 0 cadena de texto1 es menor que cadena de texto2

Comentarios

Ninguno.

Relativo

[StringInStr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#), [StringUpper](#)

Ejemplo

```
$result = StringCompare("MELÓN", "melón")
MsgBox(0, "Resultado de StringCompare (modo 0):", $result)
```

```
$result = StringCompare("MELÓN", "melón", 1)
MsgBox(0, "Resultado de StringCompare (modo 1):", $result)
```

```
$result = StringCompare("MELÓN", "melón", 2)
MsgBox(0, "Resultado de StringCompare (modo 2):", $result)
```

StringInStr

Revisa si una cadena contiene una subcadena dada.

StringInStr ("cadena de texto", "subcadena" [, caso_sensitivo [, ocurrencia [, inicio [, cantidad]]]])

Parámetros

cadena de texto	La cadena a evaluar.
-----------------	----------------------

subcadena	La subcadena a buscar.
caso_sensitivo	<p>[opcional] Bandera para indicar si las operaciones deberían ser caso sensitivo. 0 = no es caso sensitivo, usando el usuario local (por defecto) 1 = caso sensitivo 2 = no es caso sensitivo, usando una comparación básica/rápida</p>
ocurrencia	[opcional] Cual ocurrencia de la subcadena a buscar en la cadena. Usa ocurrencia negativa para buscar del lado derecho. El valor predefinido es 1 (busca la primer ocurrencia).
inicio	[opcional] El inicio de posición de búsqueda.
cantidad	[opcional] El número de caracteres a buscar. Esto efectivamente limita la búsqueda a una porción de toda la cadena. Ver Comentarios.

Valor de Retorno

Con Éxito: Devuelve la posición de la subcadena.

Al Fallar: Devuelve 0 si la subcadena no es encontrada.

@Error 0 - Operación normal

1 - Parámetro inválido de "inicio" y/o "ocurrencia".

Comentarios

El primer carácter de posición es 1.

El parámetro "cantidad" debe ser más largo que la subcadena a buscar. Este parámetro (combinado con el parámetro inicio) efectivamente limita la búsqueda de una subcadena en toda la cadena. Las dos declaraciones siguientes son equivalentes:

`StringInStr("la cadena abuscar", "cadena", 0, 1, 1, 11)`

`StringInStr(StringMid("la cadena abuscar", 1, 11), "cadena")`

Relativo

[StringCompare](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#), [StringUpper](#), [StringRegExp](#), [StringSplit](#)

Ejemplo

```
$result = StringInStr("Soy una cadena", "RING")
MsgBox(0, "resultado de búsqueda:", $result)
$location = StringInStr("¿Cuánta madera pondría maderalo un plato en un plato de madera
podría maderalo?", "madera", 0, 3) ; Busca la 3ra ocurrencia de "madera"
```

StringIsAINum

Revisa si una cadena contiene solamente caracteres alfanuméricos.

StringIsAINum ("cadena de texto")

Parámetros

cadena de texto	La cadena a revisar
-----------------	---------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la cadena contiene caracteres no-alfanuméricos.

Comentarios

Tenga presente que cualquier espacio en blanco causará que StringIsAINum devuelva 0.

Relativo

[StringIsAlpha](#), [StringIsASCII](#), [StringIsDigit](#), [StringIsLower](#), [StringIsSpace](#), [StringIsUpper](#),
[StringIsXDigit](#), [StringUpper](#), [StringLower](#)

Ejemplo

```
$x = "Esta es una frase con espacios en blanco."
MsgBox(0, "StringIsAINum devuelve:", StringIsAINum($x))
```

StringIsAlpha

Revisa si una cadena contiene solamente caracteres alfabéticos.

StringIsAlpha ("string")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la cadena contiene Caracteres no-alfabéticos

Comentarios

Tenga presente que cualquier espacio en blanco causará que StringIsAlpha devuelva 0.

Relativo

[StringIsAINum](#), [StringIsASCII](#), [StringIsDigit](#), [StringIsLower](#), [StringIsSpace](#), [StringIsUpper](#),
[StringIsXDigit](#), [StringUpper](#), [StringLower](#)

Ejemplo

```
$x = "Esta es una frase con espacios en blanco."
MsgBox(0,"StringIsAlpha devuelve:", StringIsAlpha($x))
```

StringIsASCII

Revisa si una cadena contiene solamente caracteres ASCII en el rango 0x00 - 0x7f (0 - 127).

StringIsASCII ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la cadena contiene caracteres extendidos de ASCII.

Comentarios

Ninguno.

Relativo

[StringIsAINum](#), [StringIsAlpha](#), [StringIsDigit](#), [StringIsLower](#), [StringIsSpace](#), [StringIsUpper](#),
[StringIsXDigit](#), [StringUpper](#), [StringLower](#)

Ejemplo

```
$x = StringIsASCII("Esta es una frase.")  
MsgBox(0,"StringIsASCII devuelve:", $x)
```

StringIsDigit

Revisa si una cadena contiene solamente caracteres de dígitos (0-9).

StringIsDigit ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la cadena contiene no-dígitos.

Comentarios

Ver ejemplos.

Relativo

[StringIsAINum](#), [StringIsAlpha](#), [StringIsASCII](#), [StringIsLower](#), [StringIsSpace](#), [StringIsUpper](#),
[StringIsXDigit](#), [StringIsInt](#)

Ejemplo

```
StringIsDigit("12333") ;devuelve 1  
StringIsDigit("1.5") ;devuelve 0 debido al punto decimal  
StringIsDigit("1 2 3") ;devuelve 0 debido al espacio en blanco  
StringIsDigit("") ;devuelve 0
```

StringIsFloat

Revisa si una cadena es un número tipo punto flotante.

StringIsFloat ("cadena de texto")

Parámetros

cadena de texto	La cadena o expresión a revisar.
-----------------	----------------------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si no contiene el formato de punto flotante.

Comentarios

Una cadena es flotante si contiene como mínimo un dígito decimal y exactamente un punto; el solamente hecho de otro carácter permitido es opcional, el signo más o menos al empezar. (StringIsFloat no acepta un carácter de coma como el punto decimal aún si la configuración local de la computadora usa ese símbolo.) Si el parámetro no es una cadena de texto, su valor es convertido a una cadena. Ver ejemplos.

Relativo

[StringIsInt](#), [IsFloat](#)

Ejemplo

```
StringIsFloat("1.5") ;devuelve 1  
StringIsFloat("7.") ;devuelve 1 desde que contiene una expresión decimal  
StringIsFloat("-.0") ;devuelve 1  
StringIsFloat("3/4") ;devuelve 0 desde que por la pléca en '3' y '4' no es un flotante  
StringIsFloat("2") ;devuelve 0 desde que '2' es un entero, no un flotante
```

```
StringIsFloat(1.5) ;devuelve 1 desde que 1.5 convertido a cadena lo contiene.  
StringIsFloat(1.0) ;devuelve 0 desde que 1.0 convertido a cadena no lo contiene.
```

StringFormat

Devuelve una cadena de texto con formato (similar a la función sprintf() de C)

StringFormat ("control de formato", var1 [, ... var32])

Parámetros

control de formato	El formato y banderas para usar (ver Observaciones)
var1...var32	Hasta 32 variables que será la salida de acuerdo al "control de formato".

Valor de Retorno

Devuelve la cadena de texto con formato de acuerdo a una "variable formato" definido en el parámetro de "control de formato".

Comentarios

Para prevenir una sobrecarga de memoria temporal, cada "variable" es limitada a 65535 caracteres.

Caracteres de Escape pueden contener en el "control de formato" tales como \n (@LF), \r (@CR), \t (@TAB]. Así que si usted desea tener un "\" usted necesita usar \\, lo mismo para "%" %%.

"formato de variable" es; %[banderas] [ancho] [.precisión] tipo
Especificación de ancho

El segundo campo opcional de la especificación de formato es el ancho. El argumento de ancho es un decimal positivo controlando el número mínimo de caracteres impresos. Si el número de caracteres en la salida es menor que el ancho especificado, espacios en blanco son añadidos a la izquierda o la derecha de los valores — dependiendo si la – bandera (para alinear a la izquierda) es especificado — hasta que el ancho mínimo es alcanzado. Si el ancho es prefijado con 0, los ceros son añadidos hasta que el ancho mínimos es alcanzado (no es útil para números alineados a la izquierda).

La especificación del ancho nunca causa un valor para ser truncado. Si el número de caracteres en la salida es mayor que el ancho especificado, o si el ancho no es dado, todos los caracteres del valor son impresos (sujeto a la precisión de la especificación).

Tipo de especificación

Tipo	Tipo de Variable	Formato de salida
d, i	Entero	Entero decimal con signo.
o	Entero	Entero octal sin signo.
u	Entero	Entero decimal sin signo.
x	Entero	Entero hexadecimal sin signo, usando "abcdef".
X	Entero	Entero hexadecimal

		sin signo, usando "ABCDEF".	
e	Flotante		Valor con signo teniendo la forma [-]d.dddd e [sign]ddd donde d es un simple dígito decimal, dddd es uno o más dígitos decimales, ddd es exactamente tres dígitos decimales, y sign es + o -.
E	Flotante		Idéntico al formato e excepto que E en lugar de e introduce el exponente.
f	Flotante		Valor con signo con la forma [-]dddd.dddd, donde dddd es uno o más dígitos decimales. El número de dígitos antes del punto decimal depende en la magnitud del número, y el número de dígitos después del punto decimal depende en la precisión solicitada.
g	Flotante		Valor con signo impreso en formato f o e, lo que es más compacto por el valor dado y precisión. El formato e es usado solamente cuando el exponente del valor es menos que -4 o mayor o igual que el argumento de precisión. Ceros adelante son truncados, y el punto decimal solamente si una o más dígitos lo siguen.
G	Flotante		Idéntico al formato g, excepto que E, en lugar de e, introduce el exponente (donde es apropiado).
s	Cadena de texto	Cadena de texto.	

Bandera de especificación

Bandera	Significado	Predeterminado
-	Alinea a la izquierda el resultado dentro del ancho del campo dado.	Alineación a la derecha.
+	Prefija la salida con un signo (+ o -) si el valor de la salida es con un tipo de signo.	Signo aparece solamente para valores con signo negativo (-).
0	Si el <i>ancho</i> es prefijado con 0, los ceros son agregados hasta que el ancho mínimo es alcanzado. Si 0 y - aparecen, el 0 es ignorado. Si 0 es especificado con un formato entero (i, u, x, X, o, d) el 0 es ignorado.	Sin rellenos.
Blank	Prefija el valor de salida con un espacio en blanco si está con signo y es positivo; el espacio en blanco es ignorado si ambos el espacio y la bandera + aparece.	No aparece en blanco.
#	Cuando es usado con el formato o, x, o X, la bandera # prefija cualquier salida no-cero con 0, 0x, o 0X, respectivamente.	No aparece en blanco.
#	Cuando es usado con el formato e, E, o f, la bandera # fuerza la salida a que contenga un punto decimal en todos los casos.	Punto decimal aparece solamente si dígitos lo siguen.
#	Cuando es usado con el formato g o G, la bandera #	Punto decimal aparece

	fuerza el valor de salida a que contenga un punto decimal en todos los casos y previene el truncamiento de ceros antecediendo. Ignorado cuando usado c, d, i, u, o s.	solamente si dígitos los siguen. Ceros antes son truncados.
--	---	---

Especificación de Precisión

El tercer campo opcional de la especificación de formato es la especificación de la precisión. Especifica un entero decimal positivo, precedido de un punto (.), que especifica el número de caracteres para ser impresos, el número de posición decimal, o el número de dígitos significantes (ver tabla de abajo). A diferencia de la especificación del ancho, la especificación de la precisión puede causar también truncamiento del valor de salida o redondeo de un valor de punto flotante. Si la precisión es especificada como 0 y el valor para ser convertido es 0, el resultado no es la salida de caracteres, como es mostrado abajo:

```
StringFormat( "%.0d", 0 ); /* No devuelve caracteres */
```

Cómo los valores de precisión afectan el tipo

Tipo	Significado	Predeterminado
d, i, u, o, x, X	La precisión especifica el número mínimo de dígitos para ser impresos. Si el número de dígitos en el argumento es menor que la precisión, el valor de salida es rellenada con ceros a la izquierda. El valor no es truncado cuando el número de dígitos excede la precisión.	Precisión determinada es 1.
e, E	La precisión especifica el número de dígitos para ser impresos después del punto decimal. El último dígito impreso es redondeado.	Precisión predeterminada es 6; si la precisión es 0 o el punto (.) aparece sin un número después, no se imprimirá el punto decimal.
f	El valor de precisión especifica el número de dígitos después del punto decimal. Si un punto decimal aparece, como mínimo un dígito aparece antes. El valor es redondeado al número apropiado de dígitos.	La precisión predeterminada es 6; si la precisión es 0, o si el punto (.) aparece sin números después de él, el punto decimal no será impreso.
g, G	La precisión especifica el número máximo de dígitos a ser impresos.	Seis significantes dígitos son impresos, con cualquier antecedente de ceros son truncados.
s	La precisión especifica el número máximo de caracteres a ser impresos. Caracteres en exceso de precisión no son impresos.	Los caracteres son impresos hasta que un carácter nulo es encontrado.

Relativo

Ninguno.

Ejemplo

```

$n = 43951789;
$u = -43951789;

; note que el doble %%, imprime el carácter '%' literalmente
printf("%%d = '%d'\n", $n);      '43951789'      representación de un entero estándar
printf("%%e = '%e'\n", $n);      '4.395179e+007'  notación científica
printf("%%u = '%u'\n", $n);      '43951789'      entero sin signo representación de un
positivo entero
printf("%%u <0 = '%u'\n", $u);    '4251015507'    entero sin signo representación de un
negativo entero
printf("%%f = '%f'\n", $n);      '43951789.000000' representación de un punto flotante
printf("%%.2f = '%.2f'\n", $n);   '43951789.00'    representación de un punto flotante con 2
dígitos después del punto decimal
printf("%%o = '%o'\n", $n);      '247523255'    representación octal
printf("%%s = '%s'\n", $n);      '43951789'    representación de cadena
printf("%%x = '%x'\n", $n);      '29ea6ad'    representación hexadecimal (minúsculas)
printf("%%X = '%X'\n", $n);      '29EA6AD'    representación hexadecimal (mayúsculas)

printf("%%+d = '%+d'\n", $n);    '+43951789'    especificador en un signo positivo
printf("%%+d <0= '%+d'\n", $u);  '-43951789'    especificador en un signo negativo

$s = 'mono';
$t = 'muchos monos';

printf("%%s = [%s]\n", $s);      [mono]      salida estándar de cadena
printf("%%10s = [%10s]\n", $s);   [ mono]    justificación a la derecha con espacios
printf("%%-10s = [%-10s]\n", $s);  [mono ]   justificación a la izquierda con espacios
printf("%%010s = [%010s]\n", $s);  [0000mono] cero-relleno trabaja en cadenas también
printf("%%10.10s = [%10.10s]\n", $t); [muchos mono] justificación a la izquierda pero con
un corte de 10 caracteres

printf("%04d-%02d-%02d\n", 2008, 4, 1);

Func Printf($format, $var1, $var2=-1, $var3=-1)
If $var2=-1 Then
  ConsoleWrite(StringFormat($format, $var1))
Else
  ConsoleWrite(StringFormat($format, $var1, $var2, $var3))
EndIf
EndFunc

```

StringFromASCIIArray

Convierte un arreglo de códigos ASCII a una cadena.

StringFromASCIIArray(arreglo, [inicio [, final [, codificación]]])

Parámetros

arreglo	El arreglo de códigos ASCII para convertir a caracteres.
inicio	[optional] El índice de base 0 desde donde iniciar el procesamiento (Por defecto: 0)
final	[optional] El índice de base 0 hasta donde se procesará (por defecto: UBound(\$array) - 1).
codificación	[optional] El arreglo contendrá valores en los siguientes conjuntos de caracteres: 0 - UTF-16 (por defecto) 1 - ANSI 2 - UTF-8

Valor de Retorno

1 - La entrada no es un arreglo.

2 - Índice de inicio no válido.

Comentarios

La cadena devuelta puede contener caracteres Chr(0) embebidos pero estos no serán aplicados sobre la misma. Muchas funciones de cadena se detendrán ante al primer Chr(0) encontrado, sin embargo, si es necesario el acceso al contenido completo de la cadena entonces la función StringToBinary() puede convertirla en una cadena binaria preservando todos los datos.

Si usted intenta crear un arreglo manualmente (En oposición a usar una rreglo devuelto con StringToASCIIArray()) entonces los códigos en el arreglos deben ser espesificados en UNICODE.

Relativo

[StringToASCIIArray](#)

Ejemplo

```
#include <Array.au3> ; Para_ArrayDisplay()

; Convierte una cadena en un areglo.
Local $a = StringToASCIIArray("abc")

; Muestra el arreglo para ver los valores de códigos ASCII
; de cada caracter.
_ArrayDisplay($a)

; Ahora convierte el arreglo en una cadena.
```

```
Local $s = StringFromASCIIArray($a)  
;  
; Muestra la cadena para ver si concuerda con la cadena original.  
MsgBox(0, "", $s)
```

StringIsInt

Revisa si una cadena es un entero.

StringIsInt ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la cadena no puede ser un entero.

Comentarios

StringIsInt también devuelve 1 para expresiones que no son cadenas; sin embargo, StringIsInt devuelve 0 para expresiones hexadecimales tales como "4ff0". La puntuación permitida sólo es una más o un menos en el principio de la cadena.

Relativo

[StringIsFloat](#), [StringIsDigit](#), [IsInt](#)

Ejemplo

```
StringIsInt("+42") ;devuelve 1  
StringIsInt("-00") ;devuelve 1  
StringIsInt("1.0") ;devuelve 0 debido al punto decimal  
StringIsInt(1.0) ;devuelve 1 debido a la conversión número-cadena  
StringIsInt("1+2") ;devuelve 0 debido al signo mas
```

StringIsLower

Revisa si una cadena contiene solamente letras minúsculas.

StringIsLower ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si la cadena contiene caracteres que no sean letras minúsculas.

Comentarios

Note que dígitos/puntuación/espacios en blanco causarán que StringIsLower devuelva 0.

Relativo

[StringIsAINum](#), [StringIsAlpha](#), [StringIsASCII](#), [StringIsDigit](#), [StringIsSpace](#), [StringIsUpper](#),
[StringIsXDigit](#), [StringUpper](#), [StringLower](#)

Ejemplo

```
If StringIsLower("abcfoo") Then  
    MsgBox(0, "Resultado:", "Cadena contenida sólo en letras minúsculas")  
EndIf
```

StringIsSpace

Revisa si una cadena contiene solamente caracteres de espacios en blanco.

StringIsSpace ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si cadena contiene caracteres que no son espacios en blanco.

Comentarios

Espacios en blanco incluye Chr(9) por Chr(13) que son HorizontalTab, LineFeed, VerticalTab, FormFeed, y CarriageReturn.

Espacios en blanco también incluye el espacio estándar (Chr(32)).

Relativo

[StringIsAINum](#), [StringIsAlpha](#), [StringIsASCII](#), [StringIsDigit](#), [StringIsLower](#), [StringIsUpper](#),
[StringIsXDigit](#), [StringUpper](#), [StringLower](#), [StringStripCR](#), [StringStripWS](#)

Ejemplo

```
$x = " " & @LF & Chr(11) & @TAB & " " & @CRLF
If StringIsSpace($x) Then
    MsgBox(0, "", "Cadena contiene caracteres con espacios en blanco.")
EndIf
```

StringIsUpper

Revisa si una cadena contiene solamente letras mayúsculas.

StringIsUpper ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si cadena contiene letras que no son mayúsculas.

Comentarios

Note que dígitos/puntuación/espacios en blanco causarán que StringIsUpper devuelva 0.

Relativo

[StringIsAINum](#), [StringIsAlpha](#), [StringIsASCII](#), [StringIsDigit](#), [StringIsLower](#), [StringIsSpace](#),
[StringIsXDigit](#), [StringUpper](#), [StringLower](#)

Ejemplo

```
If StringIsUpper("ABCDF") Then
```

```
MsgBox(4096, "", "Cadena contiene sólo letras MAYUSCULAS")
EndIf
```

StringIsXDigit

Revisa si una cadena contiene solamente dígitos (0-9) y caracteres hexadecimales (A-F)

StringIsXDigit ("cadena de texto")

Parámetros

cadena	La cadena a revisar.
--------	----------------------

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0 si cadena contiene caracteres no-hexadecimales.

Comentarios

Caracteres válidos son **0123456789abcdefABCDEF** Note que una cadena conteniendo espacios en blanco o el prefijo "0x" causará StringIsXDigit devuelva 0.

Relativo

[StringIsAINum](#), [StringIsAlpha](#), [StringIsASCII](#), [StringIsDigit](#), [StringIsLower](#), [StringIsSpace](#),
[StringIsUpper](#), [StringUpper](#), [StringLower](#)

Ejemplo

```
StringIsXDigit("00FC") ;devuelve 1
StringIsXDigit("2570") ;devuelve 1
StringIsXDigit("a cafe") ;devuelve 0 debido al espacio
StringIsXDigit(1 + 2.0) ;devuelve 1 debido a la conversión de número-cadena
```

StringLeft

Devuelve un número de caracteres a partir del lado izquierdo de una cadena de texto.

StringLeft ("cadena de texto", contador)

Parámetros

cadena	La cadena a evaluar.
contador	El número de caracteres a extraer.

Valor de Retorno

Devuelve una cadena conteniendo la cantidad de caracteres especificada a partir de la *izquierda* de la cadena.

Comentarios

Si *contador* excede la longitud de la cadena, la cadena entera es devuelta.

Si *contador* es negativa, una cadena vacía es devuelta.

Relativo

[StringInStr](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#),
[StringUpper](#), [StringCompare](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$result = StringLeft("Soy una cadena", 3)
MsgBox(0, "3 caracteres de la izquierda son:", $result)
```

StringLen

Devuelve el número de caracteres en una cadena de texto.

StringLen ("cadena de texto")

Parámetros

cadena	La cadena a evaluar.
--------	----------------------

Valor de Retorno

Devuelve la longitud de la cadena de texto.

Comentarios

Una expresión numérica es automáticamente evaluada y convertida a cadena.

Relativo

[StringInStr](#), [StringLeft](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#),
[StringUpper](#), [StringCompare](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$len = StringLen("Cuán largo soy?")
MsgBox(0, "Largo de cadena es:", $len)
```

StringLower

Convierte a cadena a minúsculas.

StringLower ("cadena de texto")

Parámetros

string	La cadena para convertir.
--------	---------------------------

Valor de Retorno

Devuelve la cadena convertida a minúsculas.

Comentarios

Ninguno.

Relativo

[StringIsLower](#), [StringIsUpper](#), [StringInStr](#), [StringLeft](#), [StringLen](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#), [StringUpper](#), [StringCompare](#), [StringIsAINum](#), [StringIsAlpha](#), [StringIsASCII](#), [StringIsSpace](#), [StringIsXDigit](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$var = StringLower("Soy una cadena")
MsgBox(0, "Cadena convertida a minúscula es:", $var)
```

StringMid

Extrae un número de caracteres de una cadena de texto.

StringMid ("cadena de texto", iniciar [, contador])

Parámetros

cadena	La cadena a evaluar.
--------	----------------------

inicio	La posición de carácter a iniciar. (1 = primer carácter)
contador	[opcional] El número de caracteres a extraer. Por defecto todo el resto de la cadena.

Valor de Retorno

Devuelve la cadena extraída.

Comentarios

Si *inicio* está fuera-de-los-límites, una cadena vacía es devuelta. Si la posición de *inicio* es válida pero el contador está fuera de los límites, todo el resto será devuelto.

Relativo

[StringInStr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#),
[StringUpper](#), [StringCompare](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$var = StringMid("Soy una cadena de texto", 3, 2)
MsgBox(0, "2 caracteres extraídos de la posición 3 son:", $var)
```

StringRegExp

Revisa si una cadena encaja en un patrón dado de expresión regular.

StringRegExp ("prueba", "patrón" [, flag] [, compensar])

Parámetros

prueba	La cadena a revisar.
patrón	La expresión regular a comparar.
flag	[opcional] Un número para indicar cómo la función se comportará. Ver debajo para detalles. El predefinido es 0.
compensar	[opcional] La posición de cadena a iniciar la coincidencia (inicia en 1) El predefinido es 1.

Bandera	Valores
0	Devuelve 1 (coincide) o 0 (no coincide)
1	Devuelve un arreglo de coincidencias.
2	Devuelve un arreglo de coincidencias incluyendo la coincidencia completa (estilo Perl / PHP)
3	Devuelve un arreglo de coincidencias globales.
4	Devuelve un arreglo de arreglos conteniendo coincidencias globales incluyendo la coincidencia completa (estilo Perl / PHP)

Valor de Retorno

Bandera = 0 :

@Error	Significado
2	Patrón malo. @Extended = compensación de error en el patrón.

Bandera = 1 o 2 :

@Error	Significado
0	Arreglo es válido. Revisa @Extended para la próxima compensación
1	Arreglo no es válido. No hay coincidencias.
2	Patrón malo, el arreglo no es válido. @Extended = compensación de error en patrón.

Bandera = 3 o 4 :

@Error	Significado
0	Arreglo es válido.
1	Arreglo no es válido. No hay coincidencias.

- 2 Patrón malo, el arreglo no es válido. @Extended = compensación de error en patrón.

Comentarios

El parámetro flag puede ser uno de 5 valores (0 hasta 4). 0 asigna true (1) o false (0) como si el patrón hubiera sido encontrado o no. 1 y 2 busca la primera coincidencia y devuelve este en un arreglo. 3 y 4 buscan múltiples impactos y rellena el arreglo con todo el texto encontrado. 2 y 4 incluye el texto completo en el primer índice del arreglo, no solamente captura los grupos, todo lo cual usted obtiene con los flag 1 y 3.

La notación de expresiones regulares son una manera de especificar un patrón para las cadenas que se pueden buscar. Las expresiones regulares son cadenas de caracteres en caracteres de texto plano que indican qué texto debe existir en la cadena de destino, y algunos caracteres que dan un significado especial para indicar lo que se permite la variabilidad en la cadena objetivo. Expresiones regulares en AutoIT son normalmente entre mayúsculas y minúsculas.

Las expresiones regulares se construyen de uno o más de las siguientes expresiones regulares sencillas especificadoras. Si el carácter no está en la siguiente tabla, entonces sólo coincidirá con sí mismo.

Repetición de caracteres (*, +, ?, {...}) Tratará de igualar el mayor conjunto posible, que permite que los siguientes caracteres para igualar y, salvo que irá inmediatamente seguido de un signo y, a continuación, encontrará el modelo más pequeño que permite los siguientes caracteres para igualar también.

Grupos anidados son permitidos, pero tenga en cuenta que todos los grupos, excepto la no captura de los grupos, asignar al arreglo devuelto, con los grupos exteriores después de la asignación de grupos internos.

Descripción completa se puede encontrar [aquí](#)

(<http://www.autoitscript.com/autoit3/pcrepattern.html>)

Precaución: Las expresiones regulares malas puede producir un bucle infinito, acaparando casi la CPU y pueden producir un error grave.

Caracteres de Coincidencia

[...]	Coincide cualquier carácter en el set. Eje. [aeiou] coincide con cualquier vocal minúscula. Un contiguo establece que puede ser definido usando un guión entre el inicio y el final de caracteres. Eje. [a-z] coincide con cualquier carácter minúscula. Para incluir un guión (-) en un set, usarlo como el primer o el último de carácter del set. Para incluir un corchete cerrado en un set, usarlo como el primer carácter de el set. Eje. [][] tampoco coincidirá [o]. Note que caracteres
---------	---

	especiales no retiene sus significados especiales dentro un set, con la excepción de \\", \^, \-, \[y \] coincide con el carácter de escape dentro un set.
[^ ...]	Coincide cualquier carácter no en el set. Eje. [^0-9] coincide con cualquier no-dígito. Para incluir un caret (\^) en un set, ponerlo después de empezar de establecer o escaparlo (\^).
[:class:]	Coincide un carácter en la clase de caracteres. Clases válidas son: alpha (algún carácter alfabético), alnum (algún carácter alfanumérico), lower (alguna letra minúscula), upper (alguna letra mayúscula), dígi (algún dígito decimal 0-9), xdigit (algún dígito hexadecimal, 0-9, A-F, a-f), space (algún espacio en blanco), blank (sólo un espacio o tabulador), print (algún carácter imprimible), graph (algún carácter imprimible excepto espacios), cntrl (algún carácter de control [ascii 127 o <32]) o punct (algún carácter de puntuación). Así [0-9] es equivalente a [[:digit:]].
[^:class:]	Coincide con cualquier carácter no en la clase, pero solamente si el primer carácter.
(...)	Grupo. Los elementos en el grupo son tratados en orden y pueden ser repetidos juntos. Eje. (ab)+ coincidirá "ab" o "abab", pero no "aba". Un grupo también guardará el texto coincide para usar en copia de las referencias y en el arreglo devuelto por la función, dependiendo en el valor de bandera.
(?i)	Bandera de caso-insensitivo. Este no opera como un grupo. Dice al motor de expresión regular para hacer un caso-insensitivo coincidiendo de ese punto.
(?-i)	(por defecto) Bandera de caso-sensitivo. Este no opera como un grupo. Dice al motor de expresión regular para hacer un caso-sensitivo coincidiendo de ese punto. Grupo de caso-sensitivo. Se comporta simplemente como un grupo normal, pero realiza coincidencias de caso-sensitivo dentro del grupo. Primariamente para usar después bandera (-i) o dentro de un grupo de caso-insensitivo.
(?: ...)	Grupo no-captador. Se comporta simplemente como un grupo normal, pero no guarda los caracteres coincidentes en el arreglo tampoco puede el texto coincidido ser usado para volver a referenciar.
(?i: ...)	Grupo caso-insensitivo no-captador. Se comporta simplemente como un grupo no-captador, pero realiza caso-insensitivo coincide dentro del grupo.
(?-i: ...)	Grupo caso-sensitivo no-captador. Se comporta simplemente como un grupo no-captador, pero realiza caso-sensitivo coincide dentro del grupo.

(?m)	^ y \$ coincide con nuevas líneas dentro de datos.
(?s)	. coincide cualquier incluyendo una nueva línea. (por defecto "." no coincide con una nueva línea)
(?x)	Ignora espacios en blanco y # comentarios.
(?U)	Invertir codicia de los cuantificadores.
.	Coincide con cualquier carácter simple (excepto nueva línea).
	O. La expresión en un lado o el otro puede ser coincidido.
\	<i>Escape</i> un carácter especial (tiene coincidencia con el carácter actual) o introduce un carácter especial (ver más abajo).
\\	Coincide con una barra actual (\).
\a	Alarma, que es, el carácter BEL (chr(7)).
\A	Coincide solamente empezando de cadena.
\b	Coincide a una palabra.
\B	Coincide cuando no es una palabra.
\c	Coincide con un carácter de control, basado en el próximo carácter. Por ejemplo, \cM coincide con ctrl-M.
\d	Coincide con cualquier dígito (0-9).
\D	Coincide con cualquier no-dígito.
\e	Coincide con un carácter de escape (chr(27)).
\E	caso la modificación final.
\f	Coincide con un carácter formfeed (chr(12)).
\h	cualquier espacio horizontal.
\H	cualquier carácter que no es un espacio horizontal.

\n	Coincide una línea de alimentación (@LF, chr(10)).
\Q	patrón de cita (desactivado) metacaracteres hasta \E.
\r	Coincide un retorno de carro (@CR, chr(13)).
\s	Coincide cualquier espacio en blanco: Chr(9) a través de Chr(13) que son Tabulador Horizontal, Line Feed, Tabulador Vertical, Form Feed, y Retorno de carro, y el espacio estándar (Chr(32)).
\S	Coincide con cualquier carácter que no es espacio.
\t	Coincide con un carácter de tabulador (chr(9)).
\v	cualquier espacio vertical en blanco.
\V	cualquier carácter que no es un espacio vertical espacio en blanco.
\w	Coincide con cualquier carácter "palabra": a-z, A-Z, 0-9 o línea de subrayar (_).
\W	Coincide con cualquier carácter que no es palabra.
\###	Coincide con el carácter ascii cuyo código es dado o copias de referencia. Puede ser como máximo 3 dígitos octales. Coincide con copias de referencia si son encontradas. Coincide con el primer grupo número dado exactamente. Por ejemplo, ([:alpha:]):\1 coincidirá una letra doble.
\x##	Coincide el carácter ascii cuyo código es dado en hexadecimal. Puede ser como máximo 2 dígitos.
\z	Coincide solamente al final de la cadena.
\Z	Coincide solamente al final de la cadena, o antes de una nueva línea al final.

Caracteres de repetición

{x}	Repite el carácter anterior, en un set o un grupo exactamente x veces.
{x,}	Repite el carácter anterior, en un set o un grupo como mínimo x veces.
{0,x}	Repite el carácter anterior, en un set o un grupo en la mayoría de x veces.

{x, y}	Repite el caracter anterior, en un set o un grupo entre x y y veces, inclusive.
*	Repite el caracter anterior, en un set o un grupo 0 o más veces. Equivalente a {0,}
+	Repite el caracter anterior, en un set o un grupo 1 o más veces. Equivalente a {1,}
?	El caracter anterior, en un set o un grupo puede o puede que no aparezca. Equivalente a {0, 1}
? (después de que un caracter se repita)	Busca la más pequeño coincidencia en lugar de la más grande.

Clases de Caracter

[:alnum:]	letras y dígitos
[:alpha:]	letras
[:ascii:]	códigos caracter 0 - 127
[:blank:]	espacio o tabulador solamente
[:cntrl:]	caracter de control
[:digit:]	dígitos decimales (igual como \d)
[:graph:]	caracteres de impresión, excluyendo espacio
[:lower:]	letras minúsculas
[:print:]	caracteres de impresión, incluyendo espacio
[:punct:]	caracteres de impresión, excluyendo letras y dígitos
[:space:]	espacio en blanco (no es muy parecido como \s, incluye VT: chr(11))
[:upper:]	letras mayúsculas

[:word:]	caracteres "palabra" (igual como \w)
[:xdigit:]	dígitos hexadecimales

Relativo

[StringInStr](#), [StringRegExpReplace](#)

Ejemplo

;Opción 1, usando offset

```
$nOffset = 1
While 1
    $array = StringRegExp('<test>a</test> <test>b</test> <test>c</Test>',
'<(?!test>(.*)?</(?!test>', 1, $nOffset)

If @error = 0 Then
    $nOffset = @extended
Else
    ExitLoop
EndIf
for $i = 0 to UBound($array) - 1
    msgbox(0, "Prueba RegExp con Opción 1 - " & $i, $array[$i])
Next
WEnd
```

;Opción 2, devolución simple, estilo php/preg_match()

```
$array = StringRegExp('<test>a</test> <test>b</test> <test>c</Test>',
'<(?!test>(.*)?</(?!test>', 2)
for $i = 0 to UBound($array) - 1
    msgbox(0, "Prueba RegExp con Opción 2 - " & $i, $array[$i])
Next
```

;Opción 3, devolución global, estilo AutoIt

```
$array = StringRegExp('<test>a</test> <test>b</test> <test>c</Test>',
'<(?!test>(.*)?</(?!test>', 3)

for $i = 0 to UBound($array) - 1
    msgbox(0, "Prueba RegExp con Opción 3 - " & $i, $array[$i])
Next
```

;Opción 4, devolución simple, estilo php/preg_match_all()

```
$array = StringRegExp('F1oF2oF3o', '(F.o)*?', 4)

for $i = 0 to UBound($array) - 1
    $match = $array[$i]
```

```

for $j = 0 to UBound($match) - 1
    msgbox(0, "Prueba cRegExp con Opción 4 - " & $i & ',' & $j, $match[$j])
Next
Next

```

StringRegExpReplace

Reemplaza texto en un cadena basado en un expresión regular.

StringRegExpReemplaza ("prueba", "patrón", "reemplazo", [contador])

Parámetros

prueba	La cadena a revisar.
patrón	La expresión regular a comparar. Ver StringRegExp para un definición de patrones.
reemplazo	El texto a reemplazar en la expresión regular coincidiendo. Para insertar el grupo de texto coincidente, \0 - \9 (o \$0 - \$9) puede ser usado como referencia.
contador	[opcional] El número de veces a ejecutar el reemplazo en la cadena. El predefinido es 0. Use 0 para reemplazo global.

Valor de Retorno

- | | |
|--------|--|
| @Error | Significado |
| 0 | Ejecutado apropiadamente. Revisa @Extended por el número de reemplazos realizados. |
| 2 | Patrón inválido. @Extended = compensación de error en patrón. |

Comentarios

Para separar referencias previas (reemplazar) desde números actuales, agrúpalos con corchetes, ejemplo: "\${1}5".

Relativo

[StringRegExp](#)

Ejemplo

Test1()

Test2()

```
; Este ejemplo demuestra un reemplazo básico. Este reemplazará las vocales a,e,i,o,u por el
; carácter @
Func Test1()
    Local $sInput = "El murciélagos revoloteaba sobre el agua"
    Local $sOutput = StringRegExpReplace($sInput, "[aeiou]", "@")
    Display($sInput, $sOutput)
EndFunc ; Test1()

; El siguiente ejemplo demuestra el uso de referencias hacia atrás para cambiar la fecha.
; desde MM/DD/YYYY para DD.MM.YYYY
Func Test2()
    Local $sInput = 'algún texto1 12/31/2009 01:02:03 algún texto2' & @CRLF &_
        'algún texto3 02/28/2009 11:22:33 some text4'
    Local $sOutput = StringRegExpReplace($sInput, '(\d{2})/(\d{2})/(\d{4})', '$2.$1.$3 ')
    Display($sInput, $sOutput)
EndFunc ; Test2()

Func Display($sInput, $sOutput)
    ; Formatea la salida
    Local $sMsg = StringFormat("Input:\t%s\nOutput:\t%s", $sInput, $sOutput)
    MsgBox(0, "Resultado", $sMsg)
EndFunc ; Display()
```

StringReplace

Reemplaza subcadenas en una cadena.

**StringReplace("cadena de texto", "cadenaBuscada/inicio", "cadenaReemplazo" [, ocurrencia
[, caso_sensitivo]])**

Parámetros

cadena	La cadena a evaluar.
"cadenaBuscada/inicio"	La subcadena a buscar o la posición de carácter a iniciar el reemplazo.
cadenaReemplazo	La cadena de reemplazo.
ocurrencia	[opcional] El número de veces a reemplazar de cadenaBuscada. Use un valor negativo para reemplazar desde el lado derecho. 0 = todas las cadenas buscadas serán reemplazadas (por defecto)
caso_sensitivo	[opcional] Bandera para indicar si las operaciones deberían ser caso

	<p>sensitivo. 0 = no sea caso sensitivo, usando el user's locale (por defecto) 1 = caso sensitivo 2 = no sea caso sensitivo, usando una comparación básico/rápido</p>
--	--

Valor de Retorno

Devuelve la nueva cadena, el número de reemplazos realizados son guardados en @extended.

Comentarios

Por defecto si ocurrencia es positiva la búsqueda/reemplazo es ejecutada de derecha a izquierda. De esta forma, StringReplace("aaa", "aa", "bb") retorna "bba"

Si el método de inicio es usado, la ocurrencia y parámetro de casosensitivo son ignorados. Si la cadenaReemplazo no puede ser guardada una cadena vacía será devuelta y @error es establecido a 1.

Relativo

[StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#), [StringTrimRight](#),
[StringUpper](#), [StringStripWS](#), [StringAddCR](#)

Ejemplo

```
$text = StringReplace("esta es una línea de texto", " ", "-")
$numreplacements = @extended
MsgBox(0, "Nueva cadena de texto es", $text)
MsgBox(0, "El número de reemplazos realizados fué de", $numreplacements)
```

StringRight

Devuelve un número de caracteres del lado izquierdo de una cadena.

StringRight ("cadena de texto", contador)

Parámetros

cadena	La cadena a evaluar.
contador	El número de caracteres a obtener.

Valor de Retorno

Devuelve una cadena conteniendo según el *contador* de caracteres a la derecha de la cadena.

Comentarios

Si el *contador* excede una longitud de cadena, la cadena entera es devuelta.

Si el *contador* es negativo, una cadena vacía es devuelta.

Relativo

[StringInStr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringTrimLeft](#), [StringTrimRight](#),
[StringUpper](#), [StringCompare](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$var = StringRight("Soy una cadena", 3)
MsgBox(0, "3 caracteres más a la derecha son:", $var)
```

StringSplit

Divide una cadena en varias partes de acuerdo a un delimitador dado.

StringSplit ("cadena de texto", "delimitador" [, flag])

Parámetros

cadena	La cadena a evaluar.
delimitador	Uno o más caracteres para usar como delimitador (caso sensitivo)
flag	Cambia como las cadenas son separadas, sume valores múltiples de flags si es requerido: flag = 0 (por defecto), cada carácter en la cadena del delimitador marcará donde para dividir la cadena. flag = 1, el demilitador completo se usará para separar las cadenas. flag = 2, Deshabilita el retorno de la cantidad de partes en el primer índice del arreglo - esto construye un arreglo de índice base cero (debe usar UBound() para obtener el tamaño del arreglo en este caso)

Valor de Retorno

Devuelve un arreglo, el primer elemento (\$arreglo[0]) contiene el número de cadenas devueltas, los elementos restantes (\$arreglo[1], \$arreglo[2], etc.) contiene las cadenas delimitadas. Si flag = 2 entonces no se devuelve la cantidad en el primer índice del arreglo.

Si no hay delimitadores encontrados, @error es establecido a 1, el contador es 1 (\$arreglo[0]) y la cadena completa será devuelta (\$arreglo[1]).

Comentarios

Si usted usa una cadena vacía "" para los delimitadores, cada carácter será devuelto como un elemento.

Si el delimitador que usted desea usar es una subcadena en lugar de carácter individual, ver el ejemplo más abajo.

StringSplit es muy útil como una alternativa a StringInStr y como un medio para crear un arreglo.

Tenga cuidado en que si usted usa el macro @CRLF usted está referenciando a 2 caracteres, así que usted generará líneas en blanco extra.

Relativo

[StringInStr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#),
[StringTrimRight](#), [StringUpper](#)

Ejemplo

```
$days = StringSplit("Sun,Mon,Tue,Wed,Thu,Fri,Sat", ",")  
; $days[1] contiene "Sun" ... $days[7] contiene "Sat"
```

```
$text = "Esta\nlínea\nno contiene\nquebradas de línea estilo-C."  
$array = StringSplit($text, '\n', 1)
```

StringStripCR

Remueve todos los valor de retorno de carro (Chr(13)) de una cadena.

StringStripCR ("cadena de texto")

Parámetros

cadena	La cadena a convertir.
--------	------------------------

Valor de Retorno

Devuelve la cadena con todas las instancias del @CR carácter (Chr(13)) removido.

Comentarios

Ninguno.

Relativo

[StringAddCR](#), [StringStripWS](#), [StringIsSpace](#)

Ejemplo

```
$result = StringStripCR("Soy una cadena" & Chr(13) & Chr(10))
MsgBox(0, "Todos los caracteres de CR cortados son:", $result)
```

StringStripWS

Recorta los espacios en blanco de una cadena.

StringStripWS ("cadena de texto", flag)

Parámetros

cadena	La cadena a recortar.
flag	Bandera para indicar el tipo de recortes que debería ser realizado (agregue las banderas juntas para operaciones múltiples): 1 = recorta los espacio en blanco del inicio 2 = recorta espacio en blanco del final 4 = recorta dobles (o más) espacios entre palabras 8 = recorta todos los espacios (sobrescribe todas los demás flag)

Valor de Retorno

Devuelve la nueva cadena recortadas del espacio buscado.

Comentarios

Espacios en blanco incluye Chr(9) hasta Chr(13) que son TabHorizontal, LineFeed, TabVertical, FormFeed, y Retorno de carro. Espacios en blanco también incluye la cadena nula (Chr(0)) y el espacio estándar (Chr(32)).

Para recortar espacios simples entre palabras, use la función StringReplace.

Relativo

[StringStripCR](#), [StringIsSpace](#), [StringReplace](#)

Ejemplo

```
$text = StringStripWS(" esta es una línea de texto ", 3)
MsgBox(0, "cortado de ambos extremos", $text)
```

StringToASCIIArray

Convierte una cadena a un arreglo contenido el código ASCII de cada carácter.

StringToASCIIArray("cadena", [inicio [, final [, codificación]]])

Parámetros

"cadena"	La cadena para convertir en un arreglo de códigos ASCII.
inicio	[opcional] La posición de base cero desde donde inicial el procesamiento (por defecto: 0).
final	[opcional] La posición de base cero hasta donde se procesará (por defecto: StringLen("cadena")).
codificación	[opcional] El arreglo devuelto contendrá caracteres en el siguiente conjunto de caracteres: 0 - UTF-16 (por defecto) 1 - ANSI 2 - UTF-8

Valor de Retorno

Comentarios

La cadena puede contener Chr(0) embebidos. Estos pueden aparecer en el arreglo devuelto así como al cualquier dato pasado al mismo. El procesamiento solo se detiene cuando el final de la cadena es alcanzado o el usuario especifica el final.

El orden para convertir datos binarios para un arreglo usando esta función es convertir primero a cadena usando la función BinaryToString().

Relativo

[StringFromASCIIArray](#), [BinaryToString](#)

Ejemplo

```
#include <Array.au3> ; For _ArrayDisplay()
```

```
Local $a = StringToASCIIArray("abc")
_ArrayDisplay($a)
```

StringTrimLeft

Corta un número de caracteres del lado izquierdo de una cadena.

StringTrimLeft ("cadena de texto", contador)

Parámetros

cadena	La cadena a evaluar.
contador	El número de caracteres a cortar.

Valor de Retorno

Devuelve una cadena cortada por la cantidad definida en *contador* a partir de la izquierda.

Comentarios

Si *contador* está fuera de límites, una cadena vacía es devuelta. StringTrimLeft(\$str, \$n) es funcionalmente equivalente a StringRight(\$str, StringLen(\$str) - \$n)

Relativo

[StringInstr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimRight](#),
[StringUpper](#), [StringCompare](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$result = StringTrimLeft("Soy una cadena", 3)
MsgBox(0, "Cadena sin los 3 caracteres de la izquierda:", $result)
```

StringTrimRight

Corta un número de caracteres del lado derecho de una cadena.

StringTrimRight ("cadena de texto", contador)

Parámetros

cadena	La cadena a evaluar.
--------	----------------------

contador	El número de caracteres a cortar.
----------	-----------------------------------

Valor de Retorno

Devuelve el cadena cortada por contador de caracteres desde la derecha.

Comentarios

Si el contador está fuera de límites, una cadena vacía será devuelta.

[StringTrimRight\(\\$str, \\$n\)](#) es funcionalmente equivalente a [StringLeft\(\\$str, StringLen\(\\$str\) - \\$n\)](#)

Relativo

[StringInStr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#), [StringRight](#), [StringTrimLeft](#),
[StringUpper](#), [StringCompare](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$result = StringTrimRight("Soy una cadena", 3)
MsgBox(0, "Cadena sin los 3 caracteres de la derecha:", $result)
```

StringUpper

Convierte una cadena a mayúsculas.

StringUpper ("cadena de texto")

Parámetros

cadena	La cadena a convertir.
--------	------------------------

Valor de Retorno

Devuelve la cadena convertida a mayúsculas.

Comentarios

Ninguno.

Relativo

[StringIsLower](#), [StringIsUpper](#), [StringInStr](#), [StringLeft](#), [StringLen](#), [StringLower](#), [StringMid](#),
[StringRight](#), [StringTrimLeft](#), [StringTrimRight](#), [StringCompare](#), [StringIsAlphaNum](#), [StringIsAlpha](#),
[StringIsASCII](#), [StringIsSpace](#), [StringIsXDigit](#), [StringReplace](#), [StringSplit](#)

Ejemplo

```
$result = StringUpper("Soy una cadena de 4-palabras")
MsgBox(0, "Cadena convertida a mayúsculas:", $result)
```

Referencia de Funciones de Tiempo e Intervalo

A continuación se muestra una lista completa de las funciones de Funciones de Tiempo e Intervalo disponibles en Autolt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
Sleep	Pausa la ejecución del script.
TimerInit	Devuelve una marcaDeTiempo (en milisegundos)
TimerDiff	Devuelve la diferencia en tiempo de una llamada previa a TimerInit().

Sleep

Pausa la ejecución del script.

Sleep (retraso)

Parámetros

retraso	Cantidad de tiempo a pausar (en milisegundos)
---------	---

Valor de Retorno

Ninguno.

Comentarios

Tiempo máximo a retrasar es de 2147483647 milisegundos (24 días)

Relativo

Ninguno.

Ejemplo

Sleep(5000) ; cinco segundos

TimerInit

Devuelve una marcaDeTiempo (en milisegundos)

TimerInit ()

Parámetros

Ninguno.

Valor de Retorno

Devuelve un número de marcaDeTiempo (en milisegundos)

Comentarios

Ninguno.

Relativo

[TimerDiff](#)

Ejemplo

```
$begin = TimerInit()  
sleep(3000)  
$dif = TimerDiff($begin)  
MsgBox(0,"Diferencia de Tiempo",$dif)
```

TimerDiff

Devuelve la diferencia en tiempo de una llamada previa a TimerInit().

TimerDiff (marcaDeTiempo)

Parámetros

marcaDeTiempo	marcaDeTiempo devuelta de una llamada previa a TimerInit()
---------------	--

Valor de Retorno

Devuelve la diferencia de tiempo (en milisegundos) de una llamada previa a TimerInit()

Comentarios

Ninguno.

Relativo

[TimerInit](#)

Ejemplo

```
$begin = TimerInit()  
sleep(3000)  
$dif = TimerDiff($begin)  
MsgBox(0,"Diferencia de Tiempo",$dif)
```

Referencia de Funciones de Barra de Tareas

A continuación se muestra una lista completa de las funciones de Funciones de Barra de Tareas disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
TrayAutoPause (Option)	Script pausa cuando se da clic en el icono de la barra de tareas.
TrayCreateItem	Crea un control de itemMenu para la bandeja.
TrayCreateMenu	Crea un control de menu para el menu de bandeja.
TrayItemDelete	Elimina un control de menu/item del menú de bandeja.
TrayItemGetHandle	Devuelve el apuntador para un menú de bandeja(item).
TrayItemGetState	Obtiene el estado actual de un control.
TrayItemGetText	Obtiene el itemtexto de un control de menú/item de bandeja.
TrayItemSetOnEvent	Define un Función de usuario para ser invocada cuando un item del ícono de Autoit es clickeado.
TrayItemSetState	Establece el estado de un control menu/item en la bandeja del sistema.
TrayItemSetText	Establece el texto del item en un control menu/item.
TrayGetMsg	Consulta cualquier evento ocurrido en un menú de bandeja.
TrayIconDebug (Option)	Si está activado muestra la línea del script actual en la bandeja de iconos (tray icono) en un tip, que ayuda a depurar.
TrayIconHide (Option)	Oculta el icono de Autolt, de la bandeja de íconos (tray icono). Nota: El

	icono inicialmente aparecerá por ~750 milisegundos.
<u>TrayMenuMode (Option)</u>	Extiende el comportamiento del ícono de la bandeja/menu. Esto puede ser una combinación(adición)de los siguientes valores.
<u>TrayOnEventMode (Option)</u>	Habilita/deshabilita las notificaciones de las funciones de OnEvent para el ícono de la barra de tareas.
<u>TraySetClick</u>	Establece el modo de clic en el ícono de la bandeja - con clics del mouse se mostrará el menú del ícono.
<u>TraySetIcon</u>	Carga/establece un ícono especificado en la bandeja.
<u>TraySetOnEvent</u>	Define una función de usuario que será invocada cuando una acción especial ocurre en la bandeja.
<u>TraySetPauselcon</u>	Carga/establece un ícono de pausa en la bandeja especificado.
<u>TraySetState</u>	Establece el estado del ícono de la bandeja del sistema.
<u>TraySetToolTip</u>	(Re)Establece el texto tooltip para el ícono de bandeja.
<u>TrayTip</u>	Muestra un globo tip desde el ícono de AutoIt. (2000/XP solamente)

TrayAutoPause

Pausa el script cuando se da clic en el ícono de la barra de tareas.

0 = no pausa

1 = pausa (por defecto). Si no existe DefaultMenu la pausa no ocurrirá.

TrayCreateItem

Crea un control de itemMenu para la bandeja.

TrayCreateItem (texto [, IDmenu [, entradaMenu [, itemRadioMenu]]])

Parámetros

texto	El texto del control.
IDmenu	[opcional] Le permite crear un submenu en el menu referenciado. Si es igual a -1 será añadido 'detrás' del último item creado (configuración por defecto)
entradaMenu	[opcional] Le permite definir el número de entrada para ser creado. Las entradas son enumeradas iniciando en 0. Si es igual a -1 será añadido 'detrás' de la última entrada creada (configuración por defecto)
itemRadioMenu	[opcional] 0 (por defecto) = crear un itemMenu normal , 1 = crear un itemRadioMenu.

Valor de Retorno

Con Éxito: Devuelve el identificador (ID del control) del nuevo itemMenu.

Al Fallar: Devuelve 0.

Comentarios

Si el parámetro de 'texto' es una cadena en blanco ("") entonces un separador de línea será creada.

Por defecto, itemMenus normales con marca (no itemMenus de radio) serán automáticamente desmarcados si usted los pulsa!

Para desactivar este comportamiento use el valor '2' en [TrayMenuMode](#).

itemMenus de Radio, son automáticamente agrupados juntos y estos grupos están separados por una línea de separación o un item normal item que no es item de radio. Por defecto, un radio itemMenu pulsado será seleccionado automáticamente y todos los demás items de radio en el mismo grupo serán deseleccionados!

Para desactivar este comportamiento use el valor '8' en [TrayMenuMode](#).

Relativo

[TrayItemSetState](#), [TrayItemSetText](#), [TrayGetMsg](#), [TrayItemDelete](#), [TrayItemSetOnEvent](#)

Ejemplo

```
; ****
; * Primer ejemplo *
; ****
#NoTrayIcon
```

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán

mostrados.

```
$prefsitem = TrayCreateItem("Preferencias")
TrayCreateItem("")
$aboutitem = TrayCreateItem("Acerca")
TrayCreateItem("")
$exititem = TrayCreateItem("Exit")

TraySetState()

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            ContinueLoop
        Case $msg = $prefsitem
            MsgBox(64, "Preferencias:", "SO:" & @OSVersion)
        Case $msg = $aboutitem
            MsgBox(64, "Acerca de:", "Ejemplo-AutoIt3.")
        Case $msg = $exititem
            ExitLoop
    EndSelect
WEnd
```

Exit

```
; ****
; * Segundo ejemplo *
; ****
```

```
#Include <Constants.au3>
#NoTrayIcon
```

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán mostrados.

```
; Crear 2 menuitmes radio
$radio1 = TrayCreateItem("Radio1", -1, -1, 1)
TrayItemSetState(-1, $TRAY_CHECKED)
$radio2 = TrayCreateItem("Radio2", -1, -1, 1)
$radio3 = TrayCreateItem("Radio3", -1, -1, 1)
```

TrayCreateItem("") ; grupos menúitem de Radios pueden ser separados por una línea separadora u otra menuitem normal

```

$radio4 = TrayCreateItem("Radio4", -1, -1, 1)
$radio5 = TrayCreateItem("Radio5", -1, -1, 1)
TrayItemSetState(-1, $TRAY_CHECKED)
$radio6 = TrayCreateItem("Radio6", -1, -1, 1)

TrayCreateItem("")

$aboutitem = TrayCreateItem("Acerca")
TrayCreateItem("")
$exititem = TrayCreateItem("Salir")

TraySetState()

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            ContinueLoop
        Case $msg = $aboutitem
            MsgBox(64, "About:", "Ejemplo de Bandeja-AutoIt3 con grupos de radios y items de menú.")
        Case $msg = $exititem
            ExitLoop
    EndSelect
WEnd

Exit

```

TrayCreateMenu

Crea un control de menu para el menu de bandeja.

TrayCreateMenu ("sub/menutexto" [, IDmenu [, entradaMenu]])

Parámetros

sub/menutexto	El texto de sub/menu.
IDmenu	[opcional] Si es definido, le permite crear un submenu en el menú referenciado. Si es igual a -1 lo referirá al primer nivel del menu.
entradaMenu	[opcional] Permite definir el número de entrada para ser creada. Las entradas

son enumeradas iniciando en 0.

Valor de Retorno

Con Éxito: Devuelve el identificador (ID del control) del nuevo menú de bandeja.

Al Fallar: Devuelve 0.

Comentarios

Ninguno.

Relativo

[TrayItemSetState](#), [TrayItemSetText](#), [TrayGetMsg](#), [TrayItemDelete](#)

Ejemplo

#NoTrayIcon

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán mostrados.

```
$settingsitem = TrayCreateMenu("Configuración")
$displayitem = TrayCreateItem("Mostrar", $settingsitem)
$printeritem = TrayCreateItem("Impresora", $settingsitem)
TrayCreateItem("")
$aboutitem = TrayCreateItem("Acerca")
TrayCreateItem("")
$exititem = TrayCreateItem("Salir")
```

TraySetState()

While 1

```
$msg = TrayGetMsg()
Select
Case $msg = 0
    ContinueLoop
Case $msg = $aboutitem
    MsgBox(64,"Acerca:","Ejemplo de AutoIt3-Tray")
Case $msg = $exititem
    ExitLoop
EndSelect
WEnd
```

Exit

TrayItemDelete

Elimina un control de menu/item del menú de bandeja.

TrayItemDelete (IDcontrol)

Parámetros

ID del control	El identificador del control (ID del control) como es devuelto por una función TrayCreateItem o TrayCreateMenu .
----------------	--

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0.

Comentarios

Ninguno.

Relativo

[TrayCreateItem](#), [TrayCreateMenu](#)

Ejemplo

#NoTrayIcon

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán mostrados.

```
$settingsitem = TrayCreateMenu("Configuración")
$displayitem = TrayCreateItem("Mostrar", $settingsitem)
$printeritem = TrayCreateItem("Impresora", $settingsitem)
TrayCreateItem("")
$aboutitem = TrayCreateItem("Acerca")
TrayCreateItem("")
$exititem = TrayCreateItem("Salir")
```

TraySetState()

While 1

\$msg = TrayGetMsg()

Select

Case \$msg = 0

ContinueLoop

```

Case $msg = $aboutitem
    MsgBox(64, "Acerca:", "Ejemplo de AutoIt3-Tray")
Case $msg = $exititem
    ExitLoop
EndSelect
WEnd

Exit

```

TrayItemGetHandle

Devuelve el apuntador para un menú de bandeja (item).

TrayItemGetHandle (IDcontrol)

Parámetros

ID del control	El identificador del control(ID del control) como es devuelto por una función como TrayCreateItem o TrayCreateMenu .
----------------	--

Valor de Retorno

Con Éxito: Devuelve el ID del identificador de un control dado.

Al Fallar: Devuelve 0.

Comentarios

Para obtener el apuntador para el menú contextual de bandeja use '0' como ID del control.

Relativo

[TrayItemGetState](#), [TrayItemGetText](#)

Ejemplo

```
#include <Constants.au3>
```

```
Opt("TrayMenuMode", 1); No mostrar el menú de contexto predeterminado
```

```
Global Const $MIM_APPLYTOSUBMENUS = 0x80000000
Global Const $MIM_BACKGROUND = 0x00000002
```

```
TraySetIcon("shell32.dll", 21)
```

```
TraySetToolTip("Esto es sólo un pequeño ejemplo para mostrar los menús con colores en la
```

bandeja" & @LF & "son fáciles bajo Windows 2000 y posteriores.")

```
$OptionsMenu = TrayCreateMenu("Opciones")
$OnTopItem = TrayCreateItem("Siempre encima", $OptionsMenu)
TrayItemSetState(-1, $TRAY_CHECKED)
$RepeatItem = TrayCreateItem("Repetir siempre", $OptionsMenu)
TrayCreateItem("")
$AboutItem = TrayCreateItem("Acerca")
TrayCreateItem("")
$ExitItem = TrayCreateItem("Salir del ejemplo")
```

SetMenuColor(0, 0xEEBB99) ; valor del color BGR, '0' significa que el menú de contexto de la bandeja es llamado así mismo
SetMenuColor(\$OptionsMenu, 0x66BB99); BGR color

While 1

```
$Msg = TrayGetMsg()
```

Switch \$Msg

```
Case $ExitItem
    ExitLoop
```

```
Case $AboutItem
```

Msgbox(64, "Acerca...", "Ejemplo de menú de bandeja con color")

```
EndSwitch
```

WEnd

Exit

; Aplicar el color al menú

```
Func SetMenuColor($nMenuID, $nColor)
```

```
$hMenu = TrayItemGetHandle($nMenuID) ; Get the internal menu handle
```

```
$hBrush = DllCall("gdi32.dll", "hwnd", "CreateSolidBrush", "int", $nColor)
$hBrush = $hBrush[0]
```

```
Local $stMenuItem = DllStructCreate("dword;dword;dword;uint;dword;ptr")
```

```
DllStructSetData($stMenuItem, 1, DllStructGetSize($stMenuItem))
```

```
DllStructSetData($stMenuItem, 2, BitOr($MIM_APPLYTOSUBMENUS, $MIM_BACKGROUND))
```

```
DllStructSetData($stMenuItem, 5, $hBrush)
```

```
DllCall("user32.dll", "int", "SetMenuItemInfo", "hwnd", $hMenu, "ptr",
DllStructGetPtr($stMenuItem))
EndFunc
```

TrayItemGetState

Obtiene el estado actual de un control.

TrayItemGetState ([ID del control])

Parámetros

ID del control	[opcional] El identificador del control(ID del control) como es devuelto por una función como TrayCreateItem o TrayCreateMenu .
----------------	--

Valor de Retorno

Devuelve el estado. Ver [Tabla de TrayItemSetState](#) para sus valores.

Comentarios

Ninguno.

Relativo

[TrayItemSetState](#), [TrayItemGetHandle](#)

Ejemplo

#NoTrayIcon

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán mostrados.

```
$getitem = TrayCreateItem("Obtener estado")
TrayCreateItem("")
$aboutitem = TrayCreateItem("Acerca")
TrayCreateItem("")
$exititem = TrayCreateItem("Salir")

TraySetState()

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            ContinueLoop
        Case $msg = $getitem
            MsgBox(64,"Estado",TrayItemGetState($aboutitem))
        Case $msg = $aboutitem
            MsgBox(64,"Acerca:","Ejemplo de AutoIt3-Tray")
        Case $msg = $exititem
            ExitLoop
    EndSelect
WEnd

Exit
```

TrayItemGetText

Obtiene el itemtexto de un control de menú/item de bandeja.

TrayItemGetText (IDcontrol)

Parámetros

ID del control	El identificador del control(ID del control) como es devuelto por una función como TrayCreateItem o TrayCreateMenu .
----------------	--

Valor de Retorno

Con Éxito: Devuelve 1.

Al Fallar: Devuelve 0.

Comentarios

Ninguno.

Relativo

[TrayItemSetText](#), [TrayItemGetHandle](#)

Ejemplo

#NoTrayIcon

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán mostrados.

```
$getitem = TrayCreateItem("Obtener texto")
TrayCreateItem("")
$aboutitem = TrayCreateItem("Acerca")
TrayCreateItem("")
$exititem = TrayCreateItem("Salir")
```

```
TraySetState()
```

```
While 1
```

```
 $msg = TrayGetMsg()
```

```
Select
```

```
 Case $msg = 0
```

```
 ContinueLoop
```

```
 Case $msg = $getitem
```

```
 MsgBox(64,"Estado",TrayItemGetText($aboutitem))
```

```

Case $msg = $aboutitem
    MsgBox(64, "Acerca:", "Ejemplo de AutoIt3-Tray")
Case $msg = $exititem
    ExitLoop
EndSelect
WEnd

Exit

```

TrayItemSetOnEvent

Define un Función de usuario para ser invocada cuando un ícono de Autoit es clickeado.

TrayItemSetOnEvent (itemId, "función")

Parámetros

itemId	El identificador del ítem(itemId) devuelto por la función TrayCreateItem .
función	El nombre de la función de usuario a invocar.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

@error: 1 si "función" no es definida.

Comentarios

Las funciones OnEvent son solamente invocadas cuando TrayOnEventMode es establecido a 1 - cuando se usa este modo TrayGetMsg No es utilizado.

Dentro de la función de usuario invocada el identificador de ítem puede ser devuelto con @TRAY_ID.

Si la función es una cadena vacía "" el usuario definido previamente es deshabilitado.

Relativo

[TrayCreateItem](#), [TrayGetMsg](#), [TrayOnEventMode \(Option\)](#), [TraySetOnEvent](#)

Ejemplo

```

#NoTrayIcon

Opt("TrayOnEventMode",1)
Opt("TrayMenuMode",1) ; tray menu items por defecto (Script Paused/Exit) no será mostrado.

TraySetClick(16) ; Solo el botón secundario del mouse mostrará el menú contextual

$infoitem = TrayCreateItem("Info")
TrayItemSetOnEvent(-1,"ShowInfo")

TrayCreateItem("")

$exititem = TrayCreateItem("Exit")
TrayItemSetOnEvent(-1,"ExitScript")

TraySetState()

While 1
    Sleep(10) ; ciclo
WEnd

Exit

; Funciones
Func ShowInfo()
    Msgbox(0,"Info","Tray OnEvent Demo")
EndFunc

Func ExitScript()
    Exit
EndFunc

```

TrayItemsetState

Establece el estado de un control menu/item en la bandeja del sistema.

TrayItemsetState (ID del control, estado)

Parámetros

ID del	El identificador del control (ID del control) devuelto por una función
--------	--

control	TrayCreateItem o TrayCreateMenu .
estado	Ver el Tabla de estado más abajo.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

Tabla de estado

Estado	Valor	Comentarios
Sin cambio	0	
\$TRAY_CHECKED	1	MenuItem será seleccionado
\$TRAY_UNCHECKED	4	MenuItem será deseleccionado
\$TRAY_ENABLE	64	MenuItem será activado
\$TRAY_DISABLE	128	MenuItem será desactivado
\$TRAY_FOCUS	256	MenuItem será seleccionado
\$TRAY_DEFAULT	512	MenuItem será establecido como el itemMenu por defecto.

Los valores de estado pueden ser sumados por ejemplo \$TRAY_CHECKED + \$TRAY_DEFAULT coloca el itemMenu como marcado (checked) y con el estado predeterminado.

Para resetear/Borrar el estado del \$TRAY_DEFAULT para un itemMenu simplemente use esta función en el item con otro estado, por instancia con \$TRAY_ENABLE.

Las constantes definidas más arriba se encuentran en #include <Constants.au3>

Relativo

[TrayItemGetState](#), [TrayCreateItem](#), [TrayCreateMenu](#), [TraySetState](#)

Ejemplo

```

#include <Constants.au3>
#NoTrayIcon

Opt("TrayMenuMode",1) ; tray menu items por defecto (Script Paused/Exit) no será mostrado.

$chkitem    = TrayCreateItem("Revisarlo")
TrayCreateItem("")
$checkeditem = TrayCreateItem("Revisado")
TrayCreateItem("")
$exititem   = TrayCreateItem("Exit")

TraySetState()

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            ContinueLoop
        Case $msg = $chkitem
            TrayItemSetState($checkeditem,$TRAY_CHECKED)
        Case $msg = $exititem
            ExitLoop
    EndSelect
WEnd

Exit

```

TrayItemSetText

Establece el texto del item en un control menu/item.

TrayItemSetText (*ID del control, texto*)

Parámetros

ID del control	El identificador del control (ID del control) devuelto por una función TrayCreateItem o TrayCreateMenu .
texto	El nuevo texto para el control menu/item.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

Para cambiar el texto del un itemMenu por defecto (Script Pausado/Salir) use la constante \$TRAY_ITEM_EXIT y \$TRAY_ITEM_PAUSE como ID del control.

Relativo

[TrayItemGetText](#)

Ejemplo

```
#Include <Constants.au3>
#NoTrayIcon

Opt("TrayAutoPause",0) ; Script no será pausado cuando se haya clickeado el icono de bandeja

$valitem = TrayCreateItem("Valor:")
TrayCreateItem("")
$aboutitem = TrayCreateItem("About")

TraySetState()

TrayItemSetText($TRAY_ITEM_EXIT,"Salir de Programa")
TrayItemSetText($TRAY_ITEM_PAUSE,"Pausar Programa")

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            ContinueLoop
        Case $msg = $valitem
            TrayItemSetText($valitem,"Valor:" & Int(Random(1,10,1)))
        Case $msg = $aboutitem
            MsgBox(64,"About:","AutoIt3-Tray-sample")
    EndSelect
WEnd

Exit
```

TrayGetMsg

Consulta cualquier evento ocurrido en un menú de bandeja.

TrayGetMsg ()

Parámetros

Ninguno.

Valor de Retorno

Devuelve un evento.

El "evento" devuelto es el ID del control ID enviando el mensaje, o su evento especial (como el clic del mouse en el ícono de bandeja). O si no hay mensajes, el evento estará en 0.

IDs de Eventos

0	No hay eventos
Control ID	el ID del control enviando el mensaje
\$TRAY_EVENT_PRIMARYDOWN	el botón primario del mouse fue presionado
\$TRAY_EVENT_PRIMARYUP	el botón primario del mouse fue liberado
\$TRAY_EVENT_SECONDARYDOWN	el botón secundario del mouse fue presionado
\$TRAY_EVENT_SECONDARYUP	el botón secundario del mouse fue liberado
\$TRAY_EVENT_PRIMARYDOUBLE	el botón primario del mouse fue doblemente presionado
\$TRAY_EVENT_SECONDARYDOUBLE	el botón secundario del mouse fue doblemente presionado

Comentarios

Esta función automáticamente espera en el CPU cuando es requerido así que puede ser usado con seguridad en ciclos cortos sin ahogar todo el CPU.

Las constantes definidas más arriba se encuentran en #include <Constants.au3>

Relativo

[TrayCreateItem](#), [TrayCreateMenu](#), [TrayItemSetOnEvent](#)

Ejemplo

#NoTrayIcon

Opt("TrayMenuMode",1) ; Los ítems de menú por defecto (del Script Pausado/Salir) no serán mostrados.

```
$settingsitem = TrayCreateMenu("Configuración")
$displayitem = TrayCreateItem("Mostrar", $settingsitem)
$printeritem = TrayCreateItem("Impresora", $settingsitem)
TrayCreateItem("")
$aboutitem = TrayCreateItem("Acerca")
```

```

TrayCreateItem("")
$exititem = TrayCreateItem("Salir")

TraySetState()

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            ContinueLoop
        Case $msg = $aboutitem
            Msgbox(64,"Acerca:","Ejemplo de AutoIt3-Tray")
        Case $msg = $exititem
            ExitLoop
    EndSelect
WEnd

Exit

```

TrayIconDebug

Si está activado muestra la línea del script actual en la bandeja de iconos (tray icono) en un tip, que ayuda a depurar.

0 = no información de depuración (por defecto)
1 = muestra la información

TrayIconHide

Oculta el ícono de AutoIt, de la bandeja de íconos (tray icono). Nota: El ícono inicialmente aparecerá por ~750 milisegundos.

0 = Muestra el ícono (por defecto)
1 = Oculta el ícono

TrayMenuMode

Extiende el comportamiento del ícono de la bandeja/menu. Esto puede ser una combinación (adición) de los siguientes valores.

0 = por defecto ítems de menu (Script Pause/Exit) son anexados para el menú creado por usuario; usuario revisa si los ítems son están verificados; si usted da doble clic en el ícono de la barra del sistema entonces el id de control es devuelto con el estilo- "Default" (por defecto).

1 = Menu no por defecto.

2 = usuario crea ítems de verificación que no se automática des-chequeados si damos clic en estos.

4 = No retorna el itemMenuID que tiene el estilo-"por defecto" en el menu principal si damos clic en el ícono de la barra del sistema.

8 = desactiva el auto chequeo de grupos de botones de radio.

TrayOnEventMode

Habilita/deshabilita las notificaciones de las funciones de OnEvent para el ícono de la barra de tareas.

0 = (por defecto) deshabilitado.

1 = habilitado.

TraySetClick

Establece el modo de clic en el ícono de la bandeja - con clics del mouse se mostrará el menú del ícono.

TraySetClick (bandera)

Parámetros

bandera	0 = El menu de la bandeja nunca será mostrado con un clic del mouse 1 = Presionando el botón primario del mouse 2 = Liberando el botón primario del mouse 4 = Doble clic en el botón primario del mouse 8 = Doble clic en el botón secundario del mouse 16 =Liberando el botón secundario del mouse 32 =Doble clic en el botón secundario del mouse 64 =Pasar el mouse sobre el ícono del tray.
---------	--

Valor de Retorno

Ninguno.

Comentarios

Estas banderas NO son valores de eventos de la bandeja!

El valor por defecto es "9" - Presionando el botón primario o secundario del mouse mostrará el menú.

Relativo

[TraySetOnEvent](#)

Ejemplo

#NoTrayIcon

Opt("TrayMenuMode",1) ; tray menu items por defecto (Script Paused/Exit) no será mostrado.

```
$settingsitem = TrayCreateMenu("Settings")
$displayitem = TrayCreateItem("Display", $settingsitem)
$printeritem = TrayCreateItem("Printer", $settingsitem)
TrayCreateItem("")
$aboutitem = TrayCreateItem("About")
TrayCreateItem("")
$exititem = TrayCreateItem("Exit")
```

TraySetState()

TraySetClick(16)

While 1

```
$msg = TrayGetMsg()
Select
Case $msg = 0
    ContinueLoop
Case $msg = $aboutitem
    MsgBox(64, "About:", "AutoIt3-Tray-sample")
Case $msg = $exititem
    ExitLoop
EndSelect
WEnd
```

Exit

TraySetIcon

Carga/establece un ícono especificado en la bandeja.

TraySetIcon ([nombreArchivo [, iconID]])

Parámetros

nombreArchivo	[opcional] El nombre del archivo de ícono para ser mostrado en la bandeja.
iconID	[opcional] El identificador del ícono si el archivo contiene múltiples íconos.

Valor de Retorno

Ninguno.

Comentarios

Para restablecer el ícono al por defecto, use la función sin parámetros:

`TraySetIcon()`.

Pasando un número positivo hará referencia a la cadena del nombre del ícono.

Pasando un número negativo causa un comportamiento de "índice" en base a 1. Algunas DLL pueden tener íconos extraídos solo con números negativos.

El parámetro nombreArchivo puede ser "blank", "info", "question", "stop" o "warning" para seleccionar algunos correspondientes íconos estándares.

Relativo

[TraySetPauseIcon](#), [TraySetState](#)

Ejemplo

`#NoTrayIcon`

`Opt("TrayMenuMode",1) ;tray menu items por defecto (Script Paused/Exit) no será mostrado.`

`$exititem = TrayCreateItem("Exit")`

`TraysetState()`

`$start = 0`

`While 1`

`$msg = TrayGetMsg()`

`If $msg = $exititem Then ExitLoop`

`$diff = TimerDiff($start)`

`If $diff > 1000 Then`

`$num = -Random(0,100,1) ; negativo para usar numeración ordinal`

`ToolTip("#icon=" & $num)`

`TraySetIcon("Shell32.dll",$num)`

`$start = TimerInit()`

`EndIf`

`WEnd`

`Exit`

TraySetOnEvent

Define una función de usuario que será invocada cuando una acción especial ocurre en la bandeja.

TraySetOnEvent (specialID, "función")

Parámetros

specialID	Ver el Tabla de ID espaciales más abajo.
función	El nombre de el función de usuario a invocar.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

@error: 1 si "función" no es definida.

Comentarios

Las funciones OnEvent son utilizadas solamente cuando TrayOnEventMode es establecido a 1 - cuando estamos en este modo TrayGetMsg no es usado.

Tabla de ID espaciales

ID especial	Valor	Comentarios
\$TRAY_EVENT_SHOWICON	-3	El icono de bandeja será mostrado.
\$TRAY_EVENT_HIDEICON	-4	El icono de bandeja será ocultado.
\$TRAY_EVENT_FLASHICON	-5	El usuario provocará un pestaño intermitente en el icono.
\$TRAY_EVENT_NOFLASHICON	-6	El usuario deshabilitará un pestaño intermitente en el icono.
\$TRAY_EVENT_PRIMARYDOWN	-7	El botón primario del botón fue presionado sobre el icono de bandeja.

\$TRAY_EVENT_PRIMARYUP	-8	El botón primario del botón fue liberado sobre el ícono de bandeja.
\$TRAY_EVENT_SECONDARYDOWN	-9	El botón secundario del botón fue presionado sobre el ícono de bandeja.
\$TRAY_EVENT_SECONDARYUP	-10	El botón secundario del botón fue liberado sobre el ícono de bandeja.
\$TRAY_EVENT_MOUSEOVER	-11	El mouse se desplaza sobre el ícono de la bandeja.
\$TRAY_EVENT_PRIMARYDOUBLE	-13	El botón primario del mouse presiona doblemente sobre el ícono de bandeja.
\$TRAY_EVENT_SECONDARYDOUBLE	-14	El botón secundario del mouse presiona doblemente sobre el ícono de bandeja.

Si la función es una cadena vacía "" la función previa de usuario es desabilitada.

Las constantes definida más arriba se encuentran en #include <Constants.au3>

Relativo

[TrayOnEventMode \(Option\)](#), [TrayItemSetOnEvent](#), [TraySetClick](#)

Ejemplo

```
#include <Constants.au3>
#NoTrayIcon

Opt("TrayOnEventMode",1)
Opt("TrayMenuMode",1) ; tray menu items por defecto (Script Paused/Exit) no será mostrado.

$exit = TrayCreateItem("Exit")
TrayItemSetOnEvent(-1,"ExitEvent")

TraySetOnEvent($TRAY_EVENT_PRIMARYDOUBLE,"SpecialEvent")
TraySetOnEvent($TRAY_EVENT_SECONDARYUP,"SpecialEvent")

TraysetState()

While 1
    Sleep(10) ; ciclo
WEnd

Exit
```

```

; Funciones
Func SpecialEvent()
Select
    Case @TRAY_ID = $TRAY_EVENT_PRIMARYDOUBLE
        MsgBox(64,"SpecialEvent-Información","Botón primario del mouse doblemente
clickeado.")
    Case @TRAY_ID = $TRAY_EVENT_SECONDARYUP
        MsgBox(64,"SpecialEvent-Info","Botón secundario del mouse clickeado.")
EndSelect
EndFunc

Func ExitEvent()
Exit
EndFunc

```

TraySetPauselcon

Carga/establece un icono de pausa en la bandeja especificado.

TraySetPauselcon ([nombreArchivo [, iconID]])

Parámetros

nombreArchivo	[opcional] El nombre del archivo de ícono para ser mostrado en la bandeja del sistema.
iconID	[opcional] El identificador de ícono si el archivo contiene múltiples íconos.

Valor de Retorno

Ninguno.

Comentarios

Para restablecer el ícono de pausa al por defecto(cruz roja), use la función sin parámetros: TraySetPauselcon().

Pasando un número positivo la referencia se hará sobre el nombre del ícono. Pasando un número negativo provocará un comportamiento de "índice" en base a 1. Algunas DLL pueden tener íconos extraídos solo con números negativos.

Relativo

[TraySetIcon](#), [TraySetState](#)

Ejemplo

```

#NoTrayIcon

TraySetPauselcon("shell32.dll",12)
TraySetState()

While 1
    $msg = TrayGetMsg()
WEnd

Exit

```

TraySetState

Establece el estado del ícono de la bandeja del sistema.

TraySetState ([bandera])

Parámetros

bandera	[opcional] Una combinación de lo siguiente: 1 = Muestra el ícono de bandeja (por defecto) 2 = Destruye/Oculta el ícono de bandeja 4 = parpadea el ícono de bandeja 8 = detiene el parpadeo del ícono de bandeja 16 = Restablece el ícono a su valor por defecto (no parpadea, el texto tip por defecto)
---------	---

Valor de Retorno

Ninguno.

Comentarios

Esta función sobrescribe las opciones de "TrayIconHide" y la configuración de "#NoTrayIcon".

El ícono normal y de pausa NO son restablecidos por esta función!

Relativo

[TrayItemSetState](#), [TraySetIcon](#), [TraySetPauselcon](#)

Ejemplo

```
#NoTrayIcon
```

Opt("TrayMenuMode",1) ; tray menu items por defecto (Script Paused/Exit) no será mostrado.

```

$exititem = TrayCreateItem("Exit")

TraySetIcon("warning")
TraySetToolTip("SOS")

TraySetState() ; Muestra el icono de bandeja

$toggle = 0

While 1
    $msg = TrayGetMsg()
    Select
        Case $msg = 0
            Sleep(1000)
            If $toggle = 0 Then
                TraySetState() ; Muestra el icono de bandeja
                $toggle = 1
            Else
                TraySetState(2) ; Oculta el icono de bandeja
                $toggle = 0
            EndIF
        Case $msg = $exititem
            ExitLoop
    EndSelect

WEnd

Exit

```

TraySetToolTip

(Re)Establece el texto tooltip para el icono de bandeja.

TraySetToolTip ([texto])

Parámetros

texto	[opcional] El nuevo texto para ser mostrado como tooltip. La longitud es limitada - ver Observaciones.
-------	---

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

En Windows 2000 y posterior la extensión del tooltip es limitado a 128 caracteres.

Para restablecer el texto de Tooltip al por defecto, use la función sin parámetros:

`TraySetToolTip()`

El ToolTip es solamente cambiado cuando el ícono de bandeja es visible!

Relativo

[TrayTip](#)

Ejemplo

`#NoTrayIcon`

```
TraySetState()  
TraySetToolTip("Este es mi nuevo texto ToolTip!")
```

```
While 1  
    Sleep(10) ; ciclo  
WEnd
```

`Exit`

TrayTip

Muestra un globo tip desde el ícono de Autolt. (2000/XP solamente)

`TrayTip ("título", "texto", tiempo límite [, opciones])`

Parámetros

título	Texto que aparecerá en la parte superior del globo tip. (63 caracteres como máximo)
texto	mensaje que aparecerá en el globo tip. (255 caracteres como máximo)
tiempo límite	Un tiempo estimado (en segundos) en que el globo tip se mantendrá. (Windows establece un mínimo-máximo de 10-30 segundo, aunque esto es una estimación.)
opciones	[opcional] Ver Observaciones. 0=Sin ícono (por defecto), 1=ícono de información, 2=ícono de advertencia, 3=ícono de error

Valor de Retorno

Ninguno

Comentarios

TrayTip funciona solamente bajo Windows 2000/XP o superior.

Un Tray tip se cerrará si el ícono de bandeja de AutoIt desaparece. Consecuentemente, el TrayTip no puede aparecer si AutoItSetOption("TrayIconHide", 1) es establecido o si el usuario ha deshabilitado el globo tip a través del Registro!

Windows XP usualmente emite un sonido mostrando un globo tip. Este sonido puede ser desactivado adicionado 16 a los parámetros de *opciones* o colocando un título vacío.

Para limpiar un globo tip que ha sido borrado, invoque otro globo tip con una cadena vacía (y cualquier título)

Relativo

[MsgBox](#), [ToolTip](#), [TrayIconHide \(Option\)](#), [TraySetToolTip](#)

Ejemplo

```
TrayTip("Yo spy un título", "y yo un mensaje", 5, 1)
 MsgBox(4096, "", "Presiona OK para otro tip.")
 TrayTip("Limpiar cualquier tip del tray","",0)
 TrayTip("", "Un tray tip diferente.", 5)
 Sleep(5000)
```

Referencia de Variables y Conversiones

A continuación se muestra una lista completa de las funciones de Variables y Conversiones disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
Asc	Devuelve el código ASCII de un carácter.
AscW	Devuelve el código unicode de un carácter.
Chr	Devuelve el carácter correspondiente a un código ASCII.
ChrW	Devuelve el carácter correspondiente a un código unicode.
Assign	Asigna un dato a una variable por su nombre.
Binary	Devuelve la representación binaria de una expresión.
BinaryLen	Devuelve el número de bytes de un dato variant binario.
BinaryMid	Extrae un número de bytes de un dato variant binario.
BinaryToString	Convierte un dato variant binario en una cadena.
Dec	Devuelve la representación numérica de una cadena hexadecimal.
Eval	Devuelve el valor de una variable definida por una cadena.
Hex	Devuelve la representación de una cadena o valor binario convertido a hexadecimal.
HWnd	Convierte una expresión en un apuntador (handle) HWnd.
Int	Devuelve la representación numérica del valor entero (el número completo) de una expresión.

<u>IsAdmin</u>	Verifica que el usuario actual tenga privilegio de administrador.
<u>IsArray</u>	Chequea si una variable es de tipo arreglo.
<u>IsBinary</u>	Chequea si una variable o expresión es de tipo binaria.
<u>IsBool</u>	Chequea si una variable es de tipo booleano.
<u>IsDeclared</u>	Chequea si una variable ha sido declarada.
<u>IsDllStruct</u>	Chequea si una variable es de tipo DllStruct.
<u>IsFloat</u>	Chequea si una variable o expresión es de tipo flotante (decimal).
<u>IsHWnd</u>	Chequea si una variable es un puntero y un apuntador válido de ventana.
<u>IsInt</u>	Chequea si una variable o expresión es de tipo entero.
<u>IsKeyword</u>	Chequea si una variable es una palabra clave (Por ejemplo, Default).
<u>IsNumber</u>	Chequea si una variable es de tipo numérico.
<u>IsObj</u>	Chequea si una variable o expresión es de tipo objeto.
<u>IsPtr</u>	Chequea si una variable es de tipo puntero.
<u>IsString</u>	Chequea si una variable es de tipo cadena.
<u>MustDeclareVars (Option)</u>	Si esta opción es utilizada entonces todas las variables deben ser predeclaradas con Dim, Local o Global antes de poder ser usadas - evita la oportunidad de usar variables mal deletreadas causando bugs (errores).
<u>Number</u>	Devuelve la representación numérica de una expresión.
<u>Ptr</u>	Convierte una expresión en una variante de puntero.
<u>String</u>	Devuelve una representación de una expresión en cadena de texto.

StringToBinary	Convierte una cadena en datos binarios.
UBound	Devuelve el tamaño de un arreglo dimensionado.

Asc

Devuelve el código ASCII de un carácter.

Asc ("carácter")

Parámetros

carácter	El carácter para obtener el código. Si una cadena es usada, el código del primer carácter es tomado.
----------	--

Valor de Retorno

Devuelve el código ASCII de un carácter especificado.

Comentarios

Ver el Apéndice para una tabla completa de valores ASCII.

Asc("0") == 48, Asc("9") == 57, Asc("A") == 65, Asc("Z") == 90, Asc("a") == 97, Asc("z") == 122, etc.

Relativo

[AscW](#), [Chr](#), [ChrW](#)

Ejemplo

```
$code = Asc("A")
MsgBox(0, "Código ASCII de A:", $code)
```

AscW

Devuelve el código unicode de un carácter.

AscW ("carácter")

Parámetros

carácter	El carácter para obtener el código. Si una cadena es usada, el código del primer carácter es tomado.
----------	--

Valor de Retorno

Devuelve el código unicode de un carácter especificado.

Comentarios

Ver el Apéndice para una tabla completa de valores ASCII.

Relativo

[Asc](#), [Chr](#), [ChrW](#)

Ejemplo

```
$code = AscW("A")
MsgBox(0, "Código Unicode de A:", $code)
```

Chr

Devuelve el carácter correspondiente a un código ASCII.

Chr (código ASCII)

Parámetros

código ASCII	Un código ASCII en el rango entre 0-255 (ej., 65 devuelve la letra capital A).
--------------	--

Valor de Retorno

Con Éxitos Devuelve una cadena conteniendo la representación ASCII del código dado

Al Fallar Devuelve una cadena vacía y establece el valor de @error en 1 si el código ASCII es superior a 255.

Comentarios

Chr(48) == "0", Chr(57) == "9", Chr(65) == "A", Chr(90) == "Z", Chr(97) == "a", Chr(122) = "z", etc.

Una tabla completa de valores ASCII se encuentra en el apéndice.

Relativo

[Asc](#), [AscW](#), [ChrW](#), [String](#)

Ejemplo

```
$text = ""  
For $i = 65 to 90  
    $text = $text & Chr($i)  
Next  
MsgBox(0, "Alfabeto de mayúsculas", $text)
```

ChrW

Devuelve el carácter correspondiente a un código unicode.

ChrW (código unicode)

Parámetros

código unicode	Un código unicode en el rango de 0-65535 (ej., 65 devuelve la letra capital A).
-------------------	---

Valor de Retorno

Con Éxitos Devuelve una cadena conteniendo la representación del código dado.

Al Fallar Devuelve una cadena vacía y establece el valor de @error en 1 si el código unicode es superior a 65535.

Comentarios

Una tabla completa de valores ASCII se halla en el apéndice.

Relativo

[Asc](#), [AscW](#), [Chr](#), [String](#)

Ejemplo

```
$text = ""  
For $i = 256 to 512  
    $text = $text & ChrW($i)  
Next  
MsgBox(0, "Caracteres Unicode del 256 al 512", $text)
```

Assign

Asigna un dato a una variable por su nombre.

Assign ("varname", "dato" [, flag])

Parámetros

varname	El nombre de la variable para asignar el dato. No puede ser un arreglo de elementos.
dato	El dato usado para ser asignado a la variable.
flag	[opcional] controla la forma de asignación (puede sumar valores conjuntamente): 0 = (por defecto) Crea la variable si es necesario 1 = Fuerza la creación entorno Local 2 = Fuerza la creación entorno Global 4 = Falla si la variable no existe

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si es deshabilitado para crear/asignar la variable.

Comentarios

Si existe necesidad de utilizar Assign() para crear/escribir para una variable, entonces en muchas situaciones, Eval() puede ser usado para leer la variable y IsDeclared() debería ser utilizado para chequear si la variable existe.

Relativo

[Eval](#), [IsDeclared](#), [Execute](#)

Ejemplo

```
Global $variable
If Assign("variable", "Hola") Then MsgBox(4096, "", $variable) ; Muestra "Hola"
```

Binary

Devuelve la representación binaria de una expresión.

Binary (expresión)

Parámetros

expresión	Una expresión para convertir en dato binario/byte.
-----------	--

Valor de Retorno

Devuelve un variant Binario.

Comentarios

Ninguno

Relativo

[Int](#), [Number](#), [IsBinary](#), [BinaryLen](#), [BinaryMid](#), [BinaryToString](#), [StringToBinary](#)

Ejemplo

```
$var = Binary("0x00204060")
; $var es de tipo binario
msgbox(0, "IsBinary", IsBinary($var))
```

BinaryLen

Devuelve el número de bytes de un dato variant binario.

BinaryLen (binario)

Parámetros

binario	El dato binario para evaluar.
---------	-------------------------------

Valor de Retorno

Devuelve la longitud del dato binario en bytes.

Comentarios

Ninguno

Relativo

[Binary](#), [BinaryMid](#)

Ejemplo

; Crea el dato binario 0x10203040

```
$binary = Binary("0x10203040")
MsgBox(0, "Length in bytes:", BinaryLen($binary) )
```

BinaryMid

Extrae un número de bytes de un dato variant binario.

BinaryMid (binario, iniciar [, contador])

Parámetros

binario	El dato binario para evaluar.
iniciar	La posición de byte para iniciar. (1 = primer byte)
contador	[opcional] El número de bytes para extraer. Por defecto extrae el resto de la cadena a partir del inicio.

Valor de Retorno

Devuelve el dato binario extraído.

Comentarios

Si *iniciar* esta fuera de límite, un variant binario vacío es devuelto. Si *inicio* es válido pero el contador esta fuera de límite, la cadena entera del dato binario es devuelto.

Relativo

[Binary](#), [BinaryLen](#)

Ejemplo

```
;Crea el dato binario 0x10203040
$binary = Binary("0x10203040")
$extract = BinaryMid($binary, 2, 2)
MsgBox(0, "2nd and 3rd bytes are", $extract)
```

BinaryToString

Convierte un dato variant binario en una cadena.

BinaryToString (expresión [, flag])

Parámetros

expresión	Una expresión para convertir en cadena.
flag	[opcional] Cambia cómo el dato binario es devuelto: flag = 1 (por defecto), dato binario tomado como ANSI flag = 2, dato binario tomado como UTF16 LittleEndian flag = 3, dato binario tomado como UTF16 BigEndian flag = 4, dato binario tomado como UTF8

Valor de Retorno

Con Éxitos: La representación en cadena del dato binario.

Al Fallar: Una cadena vacía. @error es establecido a lo siguiente:

1 - Cadena de entrada tiene 0 extensión.

2 - Cadena de entrada tiene un número impar de bytes y se esperó que fuera UTF16 (debe contener un número par de bytes para ser UTF16 válido).

Comentarios

A diferencia de String() que devuelve una representación hexadecimal de un dato binario, esta función asume que el dato binario es una cadena y la convierte apropiadamente.

Relativo

[Binary](#), [IsBinary](#), [String](#), [StringToBinary](#), [StringToASCIIArray](#)

Ejemplo

```
; Binario ANSI a cadena
$buffer = StringToBinary("Hola - ??")
MsgBox(4096, "String() representación", $buffer)
$buffer = BinaryToString($buffer)
MsgBox(4096, "BinaryToString() ANSI representación", $buffer)

; Binario UTF16-LE a cadena
$buffer = StringToBinary("Hola - ??", 2)
MsgBox(4096, "String() representación", $buffer)
$buffer = BinaryToString($buffer, 2)
MsgBox(4096, "BinaryToString() UTF16-LE representación", $buffer)

; Binario UTF16-BE a cadena
$buffer = StringToBinary("Hola - ??", 3)
MsgBox(4096, "String() representación", $buffer)
$buffer = BinaryToString($buffer, 3)
MsgBox(4096, "BinaryToString() UTF16-BE representación", $buffer)
```

```
; Binario UTF8 a cadena
$buffer = StringToBinary("Hola - ??", 4)
MsgBox(4096, "String() representación", $buffer)
$buffer = BinaryToString($buffer, 4)
MsgBox(4096, "BinaryToString() UTF8 representación", $buffer)
```

Dec

Devuelve la representación numérica de una cadena hexadecimal.

Dec ("hex")

Parámetros

hex	La cadena hexadecimal a convertir.
-----	------------------------------------

Valor de Retorno

Con Éxitos Devuelve un entero.

Al Fallar Devuelve 0 estableciendo @error a 1 si una cadena no válida es dada o si se produce un desbordamiento.

Comentarios

La función solo trabaja con números enteros de 32 bit con signo(-2147483648 a 2147483647)

Relativo

[Hex](#)

Ejemplo

```
$dec = Dec("FFFF") ;retorna el número 65535
```

Eval

Devuelve el valor de una variable definida por una cadena.

Eval (cadena)

Parámetros

cadena	cadena representando el nombre de la variable.
--------	--

Valor de Retorno

Con Éxitos Devuelve el valor de la variable.

Al Fallar Devuelve "" (cadena vacía) con @error fijado a valor diferente de 0.

Comentarios

Si le es necesario utilizar Eval() para leer una variable, en muchas situaciones Assign() debe ser usado para crear/escribir para la variable y IsDeclared() para revisar si la variable existe.

Relativo

[IsDeclared](#), [Assign](#), [Execute](#)

Ejemplo

```
Dim $a_b = 12
$S = Eval("a" & "_" & "b") ;$S es establecido a 12
```

```
$S =Eval("c") ;$S = "" y @error = 1
```

Hex

Devuelve la representación de una cadena o valor binario convertido a hexadecimal.

Hex (expresión [, longitud])

Parámetros

expresión	La expresión a convertir.
longitud	<p>[opcional] Número de caracteres para ser devueltos (por encima de 8) para enteros. Los caracteres son truncados desde la izquierda si la longitud es muy pequeña. Este parámetro es ignorado si el dato es binario.</p>

Valor de Retorno

Con Éxitos Devuelve una cadena de *longitud* de caracteres, rellenado con cero si es necesario.

devuelve el tipo binario convertido.

Especial: Devuelve "" (cadena vacía) si longitud es menor que 1.

Al Fallar Devuelve "" (cadena vacía) y fija @error a 1 si longitud es mayor que 8.

Comentarios

Esta función solo funciona con números enteros de 32 bit con signo (-2147483648 a 2147483647).

La función también establece @error a 1 si no se dan los dígitos suficientes - la cadena devuelta contendrá entonces el último dígito significativo.

Relativo

[Dec](#), [BitAND](#), [BitNOT](#), [BitOR](#), [BitRotate](#), [BitShift](#), [BitXOR](#)

Ejemplo

```
$result = Hex(1033, 4) ;devuelve "0409"
```

HWnd

Convierte una expresión en un apuntador (handle) HWND.

HWnd (expresión)

Parámetros

expresión	Una expresión para convertir en apuntador(handle) HWND.
-----------	---

Valor de Retorno

Con Éxitos - Si el valor puede ser convertido a HWND, la representación HWND será devuelto.
Al Fallar - Si el HWND no denota una ventana válida, un HWND (NULL) 0 será devuelto y @error puesto a 1.

Comentarios

Números decimales pueden no ser convertidos en HWND.

Una cadena literal no puede ser convertida a HWND debido a que no existe garantía de que una ventana tenga el mismo HWND alguna otra vez. Esto no está estrictamente prohibido, pero es un error del programador y debería ser evitado.

Relativo

[Int](#), [String](#), [Number](#), [Ptr](#)

Ejemplo

```
Run("notepad.exe")
WinWait("[CLASS:Notepad]")
Local $hWnd = WinGetHandle("[CLASS:Notepad]")
Local $sHWND = String($hWnd) ; Comvierte a cadena
WinSetState(hWnd($sHWND), "", @SW_MINIMIZE)
Sleep(5000) ; Notepad debe ser minimizado
WinClose(hWnd($sHWND))
```

Int

Devuelve la representación numérica del valor entero (el número completo) de una expresión.

Int (expresión)

Parámetros

expresión	Una expresión para convertir en entero.
-----------	---

Valor de Retorno

Con Éxitos Devuelve un entero.

Al Fallar Devuelve 0 y coloca @error en 1 si no es un entero, un decimal o una cadena.

Comentarios

La porción fraccionaria de la expresión es truncada, así Int(1.999999) devuelve 1

Int(0/0) devuelve -9223372036854775807, por si usted se lo pregunta!

Esta función realiza correcciones menores en números de punto flotante (números decimales) para acotar la impresión natural de los números de punto flotante. Por ejemplo, las expresiones de punto flotante 0.7 + 0.2 + 0.1 produce un número de punto flotante que no es 1.0 exactamente.

Int() corrige esta anomalía, sin embargo, ciertas circunstancias extremadamente raras pueden llevar a Int() a retornar un valor no esperado (La disparidad en la detección de un valor no esperado son menores que si Int() no corrigiera todos los valores).

Relativo

[Number](#), [String](#), [Round](#), [HWnd](#), [Binary](#), [Ceiling](#), [Floor](#), [Ptr](#)

Ejemplo

`$var = Int(10.793) ;$var es el entero 10`

IsAdmin

Verifica que el usuario actual tenga privilegio de administrador.

IsAdmin ()

Parámetros

Ninguno

Valor de Retorno

Con Éxitos Devuelve 1 si el usuario actual tenga privilegio de administrador.

Al Fallar Devuelve 0 si el usuario carece de privilegios de administrador.

Comentarios

Devuelve 1 bajo Windows Vista solo si es ejecutado con derechos totales de administrador (por ejemplo. #RequireAdmin ha sido utilizado, o este ha sido realmente elevado por UAC)

Relativo

[#RequireAdmin](#)

Ejemplo

`If IsAdmin() Then MsgBox(0, "", "Derechos de administrados detectados")`

IsArray

Chequea si una variable es de tipo arreglo.

IsArray (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si no es una variable de arreglo.

Comentarios

Puede ser útil para validar parámetros de arreglos/No-arreglos definido para una función de usuario (UDF).

Relativo

[IsFloat](#), [IsInt](#), [IsString](#), [IsNumber](#), [IsBool](#), [IsHWnd](#), [IsBinary](#), [IsPtr](#), [VarGetType](#)

Ejemplo

```
$pos = WinGetPos("[CLASS:Notepad]")
If IsArray($pos) Then
    MsgBox(0, "Altura de la ventana", $pos[3])
EndIf
```

IsBinary

Chequea si una variable o expresión es de tipo binaria.

IsBinary (expresión)

Parámetros

expresión	La variable o expresión a revisar.
-----------	------------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no es de tipo binario.

Comentarios

Ninguno

Relativo

[IsArray](#), [IsFloat](#), [IsInt](#), [IsString](#), [IsNumber](#), [IsBool](#), [IsHWnd](#), [Binary](#), [BinaryToString](#), [StringToBinary](#), [VarGetType](#)

Ejemplo

```

$bin = Binary("0x00204060")
$str = "0x00204060"

msgbox(0, "IsBinary $bin", IsBinary($bin))
msgbox(0, "IsBinary $str", IsBinary($str))

```

IsBool

Chequea si una variable es de tipo booleano.

IsBool (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no es de tipo booleana.

Relativo

[IsArray](#), [IsFloat](#), [IsInt](#), [IsNumber](#), [IsString](#), [IsHWnd](#), [IsBinary](#), [IsPtr](#), [VarGetType](#)

Ejemplo

```

$b = true
If IsBool($b) Then MsgBox(0,"Satisfactorio", "$b es una variable booleana")

```

IsDeclared

Chequea si una variable ha sido declarada.

IsDeclared (expresión)

Parámetros

expresión	cadena representando el nombre de la variable para ser chequeada.
-----------	---

Valor de Retorno

Con Éxitos Devuelve 1 para variables Global o variables declaradas fuera de función alguna.

Especial: -1 para variables locales.

Al Fallar Devuelve 0 cuando la variable no puede ser hallada.

Comentarios

Si usted necesita usar IsDeclared() para revisar si existe una variable, entonces en muchas situaciones Assign() puede ser usado para crear/escribir la variable y Eval() puede ser usado para leer desde la variable.

Relativo

[Assign](#), [Eval](#)

Ejemplo

```
If Not IsDeclared ("a") then  
    MsgBox(0,"", "$a NO esta declarada") ; $a nunca ha sido asignado  
EndIf
```

\$a=1

```
If IsDeclared ("a") then  
    MsgBox(0,"", "$a ES declarado" ) ; pues previous $a=1 asignamiento  
EndIf
```

IsDIIStruct

Chequea si una variable es de tipo DIIStruct.

IsDIIStruct (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no de tipo DIIStruct según lo devuelve DIIStructCreate.

Comentarios

Ninguno

Relativo

[DIIStructCreate](#), [VarGetType](#)

Ejemplo

```
$struct = DIIStructCreate("char[256]")
$x = IsDIIStruct($struct)
```

IsFloat

Chequea si una variable o expresión es de tipo flotante (decimal).

IsFloat (variable)

Parámetros

variable	La variable o expresión a revisar.
----------	------------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no es de tipo flotante.

Comentarios

Ver ejemplos.

Relativo

[IsArray](#), [IsInt](#), [IsString](#), [IsNumber](#), [IsBool](#), [IsHWnd](#), [StringIsFloat](#), [IsBinary](#), [IsPtr](#), [VarGetType](#)

Ejemplo

```
IsFloat(3.14159) ;devuelve 1
IsFloat(3.000) ;devuelve 0 pues el valor es el entero 3
IsFloat(1/2 - 5) ;devuelve 1
IsFloat(1.5e3) ;devuelve 0 pues 1.5e3 = 1500
IsFloat("12.345") ;devuelve 0 pues es ua cadena
```

IsHWnd

Chequea si una variable es un puntero y un apuntador válido de ventana.

IsHWnd (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1 si la expresión es un puntero Y un apuntador de ventana válido.

Al Fallar Devuelve 0 si expresión no es un puntero O no es un apuntador de ventana válido.

Relativo

[IsArray](#), [IsFloat](#), [IsInt](#), [IsPtr](#), [IsString](#), [IsNumber](#), [IsBool](#), [GUICtrlGetHandle](#), [IsBinary](#), [VarGetType](#)

Ejemplo

```
Run("notepad.exe")
Local $hWnd = WinGetHandle("[CLASS:Notepad]")
If IsHWnd($hWnd) Then
    MsgBox(4096, "", "Es un HWND válido")
Else
    MsgBox(4096, "", "No es un HWND válido")
Endif
```

IsInt

Chequea si una variable o expresión es de tipo entero.

IsInt (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no es un entero.

Comentarios

`IsInt(7.5 - 2.5)` devuelve 1 (significa true)

Relativo

[IsArray](#), [IsFloat](#), [IsString](#), [IsNumber](#), [IsBool](#), [IsHWnd](#), [StringIsInt](#), [IsBinary](#), [IsPtr](#), [VarGetType](#)

Ejemplo

```
IsInt(-12345) ;devuelve 1  
IsInt(3.0000) ;devuelve 1  
IsInt("5432") ;devuelve 0 pues es una cadena  
IsInt(7.5 - 4.5) ;devuelve 1 pues el valor es el entero 3
```

IsKeyword

Chequea si una variable es una palabra clave (Por ejemplo, Default).

IsKeyword (variable)

Parámetros

variable	La variable a revisar.
----------	------------------------

Valor de Retorno

Con Éxitos Devuelve 1 si la variable es una palabra clave.

Al Fallar Devuelve 0 si variable no es una palabra clave.

Relativo

[Default](#), [VarGetType](#)

Ejemplo

```
$a = default  
If IsKeyword($a) Then MsgBox(0, "Ok", "Si que lo es!")
```

IsNumber

Chequea si una variable es de tipo numérico.

IsNumber (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no es de tipo numérico.

Comentarios

Variant puede ser de dos tipos bases: Cadena y Números. Adicionalmente, un número puede ser entero o fraccional/punto flotante.

Relativo

[IsArray](#), [IsFloat](#), [IsInt](#), [IsString](#), [IsBool](#), [IsHWnd](#), [IsBinary](#), [IsPtr](#)

Ejemplo

IsNumber(42)

IsObj

Chequea si una variable o expresión es de tipo objeto.

IsObj (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1 si expresión es una variable de tipo objeto.

Al Fallar Devuelve 0 si expresión no es una variable de tipo objeto.

Relativo

[ObjGet](#), [ObjCreate](#), [GUICtrlCreateObj](#), [ObjEvent](#), [ObjName](#), [VarGetType](#)

Ejemplo

```

$oSHELL = ObjCreate("shell.application")
if not IsObj($oSHELL) then
    MsgBox(0,"Error","$oSHELL no es un Objeto.")
else
    MsgBox(0,"Error","Objeto $oSHELL creado satisfactoriamente.")
endif

```

IsPtr

Cheque si una variable es de tipo puntero.

IsPtr (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1 si expresión es de tipo puntero.

Al Fallar Devuelve 0 si expresión no es de tipo puntero.

Relativo

[IsArray](#), [IsFloat](#), [IsHWnd](#), [IsInt](#), [IsString](#), [IsNumber](#), [IsBool](#), [Ptr](#)

Ejemplo

```

Run("notepad.exe")
Local $hWnd = WinGetHandle("[CLASS:Notepad]")
If IsPtr($hWnd) Then
    MsgBox(4096, "", "Es un Ptr válido")
Else
    MsgBox(4096, "", "No es un Ptr válido")
EndIf

```

IsString

Chequea si una variable es de tipo cadena.

IsString (variable)

Parámetros

variable	La variable/expresión a revisar.
----------	----------------------------------

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si expresión no es de tipo cadena.

Relativo

[IsArray](#), [IsFloat](#), [IsInt](#), [IsNumber](#), [IsBool](#), [IsHWnd](#), [IsBinary](#), [IsPtr](#), [String](#), [VarGetType](#)

Ejemplo

```
$x = IsString("foo")
```

MuscDeclareVars

Si esta opción es utilizada entonces todas las variables deben ser predeclaradas con Dim, Local o Global antes de poder ser usadas - evita la oportunidad de usar variables mal deletreadas causando bugs (errores).

1 = Las variables deben ser predeclaradas

0 = Las variables no necesitan ser predeclaradas (por defecto)

Number

Devuelve la representación numérica de una expresión.

Number (expresión)

Parámetros

expresión	Una expresión a convertir en un número.
-----------	---

Valor de Retorno

Devuelve un número.

Comentarios

Una cadena empezando con letras tiene un valor numérico de cero. Una cadena empezando con dígitos que no son caracteres numéricos son cortados.

Relativo

[Int](#), [String](#), [Binary](#), [Ceiling](#), [Floor](#), [HWnd](#), [Ptr](#), [Round](#)

Ejemplo

```
$w = Number(1+2+10) ;devuelve 13  
$x = Number("3.14") ;devuelve 3.14  
$y = Number("24/7") ;devuelve 24  
$z = Number("tmp3") ;devuelve 0
```

Ptr

Convierte una expresión en una variante de puntero.

Ptr (expresión)

Parámetros

expresión	Una expresión a convertir en una variante de puntero.
-----------	---

Valor de Retorno

Devuelve el puntero de la expresión dada.

String

Devuelve una representación de una expresión en cadena de texto.

String (expresión)

Parámetros

expresión	Una expresión a convertir en cadena de texto.
-----------	---

Valor de Retorno

Devuelve una cadena de texto.

Comentarios

La máxima extensión de la cadena es de 2147483647 caracteres (pero tenga en cuenta que no hay una línea de un script AutoIt que acceda más de los 4095 caracteres.)

Relativo

[Int](#), [Number](#), [IsString](#), [Chr](#), [BinaryToString](#), [ChrW](#), [HWnd](#), [Ptr](#), [StringToBinary](#)

Ejemplo

```
$var = String(10)  
;$var es la cadena de "10"
```

StringToBinary

Convierte una cadena en datos binarios.

StringToBinary (expresión [, bandera])

Parámetros

expresión	Una expresión a convertir en datos binarios.
bandera	[opcional] Cambia como la cadena es guardada como binario: bandera = 1 (por defecto), dato binario es ANSI bandera = 2, dato binario es UTF16 LittleEndian bandera = 3, dato binario es UTF16 BigEndian bandera = 4, dato binario es UTF8

Valor de Retorno

Devuelve una variante binaria.

Comentarios

Ninguno.

Relativo

[Binary](#), [BinaryToString](#), [IsBinary](#), [String](#)

Ejemplo

```
; ANSI Binario a Cadena  
$buffer = StringToBinary("Hola - ??")  
MsgBox(4096, "Representación String()", $buffer)  
$buffer = BinaryToString($buffer)  
MsgBox(4096, "Representación BinaryToString() ANSI", $buffer)  
  
; Binario UTF16-LE a Cadena
```

```

$buffer = StringToBinary("Hola - ??", 2)
MsgBox(4096, "Representación String()", $buffer)
$buffer = BinaryToString($buffer, 2)
MsgBox(4096, "Representación BinaryToString() UTF16-LE", $buffer)

; Binario UTF16-BE a cadena
$buffer = StringToBinary("Hola - ??", 3)
MsgBox(4096, "Representación String()", $buffer)
$buffer = BinaryToString($buffer, 3)
MsgBox(4096, "Representación BinaryToString() UTF16-BE", $buffer)

; Binario UTF8 a Cadena
$buffer = StringToBinary("Hola - ??", 4)
MsgBox(4096, "Representación String()", $buffer)
$buffer = BinaryToString($buffer, 4)
MsgBox(4096, "Representación BinaryToString() UTF8", $buffer)

```

UBound

Devuelve el tamaño de un arreglo dimensionado.

UBound (Arreglo [, Dimensión])

Parámetros

Arreglo	La variable de arreglo para ser interrogada.
Dimensión	[opcional] La dimensión del arreglo. Por defecto es 1, para un arreglo unidimensional. Si este parámetro es 0, el número de subscript del arreglo será devuelto.

Valor de Retorno

Con Éxitos Devuelve el tamaño de el arreglo.

Al Fallar Devuelve 0 y establece @error:

1 = Arreglo dado no es un arreglo.

2 = Dimensión del arreglo no válida

Comentarios

Recuerde que el valor devuelto por UBound es de uno más del índice último del arreglo!

Relativo

[Dim, ReDim](#)

Ejemplo

```
Dim $myArray[10][20] ;elemento 0,0 to 9,19
$rows = UBound($myArray)
$cols = UBound($myArray, 2)
$dims = UBound($myArray, 0)

MsgBox(0, "El " & $dims & "-dimensional arreglo tiene", _
$rows & " filas, " & $cols & " columnas")

;Muestra el contenido de $myArray's
$output = ""
For $r = 0 to UBound($myArray,1) - 1
    $output = $output & @LF
    For $c = 0 to UBound($myArray,2) - 1
        $output = $output & $myArray[$r][$c] & " "
    Next
Next
MsgBox(4096,"Contenido del arreglo", $output)
```

Referencia de Administración de Ventana

A continuación se muestra una lista completa de las funciones de Administración de Ventana disponibles en AutoIt. Click en una de ellas para obtener detalles de su descripción.

Consulte los ficheros de constantes incluidas en esta dirección [Ficheros de Constantes GUI Incluidas](#) si usted necesita usarlo para constantes de controles.

Función	Descripción
CaretCoordMode (Option)	Establece la forma en que son utilizadas las coordenadas, bien sea para coordenadas absolutas o relativas a la ventana actual:
ControlClick	Envía un clic de mouse a un determinado control.
ControlCommand	Envía un comando a un control.
ControlDisable	Deshabilita un control.
ControlEnable	Habilita un control.
ControlFocus	Establece el foco de entrada en un control dentro de una ventana.
ControlGetFocus	Devuelve el # de referencia del control que ha tomado el foco dentro de una ventana.
ControlGetHandle	Devuelve el apuntador interno (handle) de un control.
ControlGetPos	Devuelve el tamaño y posición de un control relativo a la ventana.
ControlGetText	Recupera el texto de un control.
ControlHide	Oculta un control.
ControlListView	Envía un comando a un control ListView32.
ControlMove	Mueve un control dentro de una ventana.

ControlSend	Envía una cadena de caracteres a un control.
ControlSetText	Establece texto de un control.
ControlShow	Muestra un control que ha sido ocultado.
ControlTreeView	Envía un comando a un control TreeView32.
StatusbarGetText	Recupera el texto de una barra de estado estándar.

CaretCoordMode

Establece la forma en que son utilizadas las coordenadas, bien sea para coordenadas absolutas o relativas a la ventana actual:

0 = coordenadas relativas a la ventana actual.

1 = coordenadas absolutas a la pantalla (por defecto).

2 = coordenadas relativas a la ventana cliente de la ventana actual.

ControlClick

Envía un clic de mouse a un determinado control.

ControlClick ("título", "texto", ID del control [, botón [, clicks [, x [, y]]]])

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controls .
botón	[opcional] El botón para clic, "izquierda", "derecho", "middle", principal, "menu", "primario", "secundario". por defecto es "izquierda".

clics	[opcional] El número de veces que se da clic con el mouse. Por defecto es 1.
x	[opcional] La posición x para el clic en el control. Por defecto es center.
y	[opcional] La posición y para el clic en el control. Por defecto es center.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0.

Comentarios

Algunos controles no responden a clic a menos que se encuentren dentro de la ventana activa. Use la función WinActivate() para forzar a la ventana a estar activa antes de utilizar ControlClick().

Usando 2 como el número de clics a enviar se logra enviar un mensaje de doble clic a un control - esto puede ser utilizado para ejecutar programas en un control explorador.

Si esta usando los botones izquierdo y derecho del mouse intercambiados, entonces el comportamiento de los botones es diferentes. "izquierdo" y "derecho" siempre dan clic sobre esos botones, aunque estos se hallan intercambiados. El botón "primario" o botón principal será el clic principal, aunque los botones se hallan intercambiados. El botón "secundario" o botón "menu" mostrará el menu contextual, aunque los botones se hallan intercambiados o no.

Botón	Normal	Intercambiado
""	izquierda	izquierda
"izquierda"	izquierda	izquierda
"middle"	Middle	Middle
"Derecha"	Derecha	Derecha
"primario"	izquierda	Right
"main"	izquierda	Derecha
"secondary"	Derecha	izquierda

"menu"	Derecha	izquierda
--------	---------	-----------

Relativo

[ControlCommand](#), [MouseClick](#), [WinActivate](#)

Ejemplo

ControlClick("[CLASS:Notepad]", "", "MDIClient1")

ControlCommand

Envía un comando a un control.

ControlCommand ("título", "texto", ID del control, "comando" [, "opciones"])

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .
comando	El comando para enviar al control.
opciones	[opcional] Parámetros adicionales requerido para algunos controles.

Valor de Retorno

Depende del comando según la tabla más abajo. En caso de un error(como un comando o ventana/control inválido), @error=1.

Comando, Opción	Valor de Retorno
"IsVisible", ""	Devuelve 1 si el Control es visible, 0 de otra manera
"IsEnabled", ""	Devuelve 1 si el control está activado, o sino 0

"ShowDropDown", ""	Despliega un ComboBox
"HideDropDown", ""	Contrae lista de un ComboBox
"AddString", 'string'	Agrega una cadena de texto al final en un ListBox o ComboBox
"DelString", <i>ocurrencia</i>	Elimina una cadena de texto de acuerdo a la ocurrencia en un ListBox o ComboBox
"FindString", 'string'	Devuelve la ocurrencia referida de la cadena de texto exacta en un ListBox o ComboBox
"SetCurrentSelection", <i>ocurrencia</i>	Fija una selección a una ocurrencia en un ListBox o ComboBox
"SelectString", 'string'	Fija una selección de acuerdo a la cadena de texto en un ListBox o ComboBox
"IsChecked", ""	Devuelve 1 si un botón está activo, de otra manera 0
"Check", ""	Activa el botón de radio o check
"UnCheck", ""	Deselecciona/desactiva un botón de radio o check
"GetCurrentLine", ""	Devuelve el número de línea donde está el cursor en un Edit
"GetCurrentCol", ""	Devuelve el número de columna donde está el cursor en un Edit
"GetCurrentSelection", ""	Devuelve el nombre del item seleccionado en un ListBox o ComboBox
"GetLineCount", ""	Devuelve la cantidad de líneas de un Edit
"GetLine", <i>línea#</i>	Devuelve el texto según el # de línea de un Edit
"GetSelected", ""	Devuelve el texto seleccionado de un Edit
"EditPaste", 'cadena'	Pega el 'cadena'(cadena de texto) de posición de un Edit
"CurrentTab", ""	Devuelve la ficha activa mostrada en un

	SystabControl32
"TabRight", ""	Se mueve a la próxima ficha a la derecha de un SystabControl32
"TabLeft", ""	Se mueve a la próxima ficha a la izquierda de un SystabControl32
"SendCommandID", <i>Command ID</i>	Simula el mensaje WM_COMMAND. Usualmente usado para controles ToolbarWindow32 - use la pestaña de la barra de herramienta de Au3Info para obtener el ID del Comando.

Comentarios

Algunos controles no responden a un clic a menos que se encuentren dentro de la ventana activa. Use la función WinActivate() para forzar a la ventana a estar activa antes de utilizar ControlClick().

Ciertos comandos trabajan en Combo y ListBox no trabajan en controles "ComboLBox".

Relativo

[ControlClick](#), [ControlDisable](#), [ControlEnable](#), [ControlFocus](#), [ControlGetPos](#), [ControlGetText](#), [ControlHide](#), [ControlMove](#), [ControlSetText](#), [ControlShow](#), [StatusbarGetText](#), [WinActivate](#), [WinMenuSelectItem](#), [WinGetClassList](#), [ControlGetFocus](#), [ControlListView](#), [ControlSend](#), [ControlTreeView](#)

Ejemplo

```
Run("notepad.exe")
ControlCommand("[CLASS:Notepad]", "", "Edit1", "GetLineCount", "")
```

ControlDisable

Deshabilita un control.

ControlDisable ("título", "texto", *ID del control*)

Parámetros

título	El título de la ventana a acceder.
--------	------------------------------------

texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve 1

Al Fallar Devuelve 0

Relativo

[ControlEnable](#), [ControlHide](#), [ControlCommand](#)

Ejemplo

```
ControlDisable("[CLASS:Notepad]", "", "MDIClient1")
```

ControlEnable

Habilita un control.

ControlEnable ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve 1

Al Fallar Devuelve 0

Relativo

[ControlDisable](#), [ControlShow](#), [ControlCommand](#)

Ejemplo

```
ControlEnable("[CLASS:Notepad]", "", "MDIClient1")
```

ControlFocus

Establece el foco de entrada en un control dentro de una ventana.

ControlFocus ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve 1

Al Fallar Devuelve 0

Relativo

[ControlGetFocus](#), [ControlCommand](#), [ControlSend](#)

Ejemplo

```
Run("notepad.exe")
ControlFocus("[CLASS:Notepad]", "", "Edit1")
```

ControlGetFocus

Devuelve el # de referencia del control que ha tomado el foco dentro de una ventana.

ControlGetFocus ("título" [, "texto"])

Parámetros

título	Título de la ventana a chequear.
texto	[opcional] Texto de la ventana a chequear.

Valor de Retorno

- Con Éxitos Devuelve el [ClassNameNN](#) del control que ha tomado el foco por teclado dentro de la ventana.
- Al Fallar Devuelve "" (cadena vacía) y establece el valor de @error en 1 si la ventana no existe.

Relativo

[ControlFocus](#), [ControlCommand](#)

Ejemplo

```
Run("notepad.exe")
$a = ControlGetFocus("[CLASS:Notepad]")
```

ControlGetHandle

Devuelve el apuntador interno (handle) de un control.

ControlGetHandle ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

- Con Éxitos Devuelve el valor del apuntador (HWND).
- Al Fallar Devuelve "" (cadena vacía) y establece el valor de @error en 1 si la ventana no coincide con el criterio.

Comentarios

Esta función devuelve un valor HWND/Handle.

Relativo

Ninguno.

Ejemplo

```
Run("notepad.exe")
$handle = ControlGetHandle("[CLASS:Notepad]", "", "Edit1")
```

ControlGetPos

Devuelve el tamaño y posición de un control relativo a la ventana.

ControlGetPos ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve un arreglo contenido el tamaño y la posición del control relativo a la ventana cliente:

 \$arreglo[0] = X posición

 \$arreglo[1] = Y posición

 \$arreglo[2] = ancho

 \$arreglo[3] = alto

Al Fallar Establece @error a 1.

Comentarios

El título/texto es referenciado a la ventana padre, de esta forma tenga cuidado con utilizar "", "" lo cual referencia a la ventana activa lo que puede no contener el ID del control requerido.

Relativo

[ControlCommand](#), [ControlMove](#)

Ejemplo

```
Run("notepad.exe")
$pos = ControlGetPos("[CLASS:Notepad]", "", "Edit1")
MsgBox(0, "Estado de ventana:", "POS: " & $pos[0] & "," & $pos[1] & " SIZE: " & $pos[2] & "," &
$pos[3] )
```

ControlGetText

Recupera el texto de un control.

ControlGetText ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve texto desde un control.

Al Fallar Establece @error a 1 y devuelve una cadena vacía "".

Relativo

[WinGetText](#), [StatusbarGetText](#), [ControlSetText](#), [ControlCommand](#)

Ejemplo

```
Run("notepad.exe")
$var = ControlGetText("[CLASS:Notepad]", "", "Edit1")
```

ControlHide

Oculta un control.

ControlHide ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana/control no es encontrado.

Relativo

[ControlShow](#), [ControlCommand](#), [ControlDisable](#), [WinSetState](#)

Ejemplo

`ControlHide("[CLASS:Notepad]", "", "MDIClient1")`

ControlListView

Envía un comando a un control ListView32.

ControlListView ("título", "texto", ID del control, "comando" [, opciones1 [, opciones2]])

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.

ID del control	El control para interactuar. Ver Controles .
comando	El comando para enviar al control (ver más abajo).
opciones1	[opcional] Parámetros adicionales requeridos por algunos comandos.
opciones2	[opcional] Parámetros adicionales requeridos por algunos comandos.

Valor de Retorno

Depende del comando de la tabla más abajo. En caso de un error (como un comando o ventana/control inválido no es encontrado) entonces @error es fijado a 1.

Comando, Opción1, Opción2	Operación
"DeSelect", From [, To]	Deselecciona uno o más ítems.
"FindItem", "cadena a buscar" [, SubItem]	Devuelve el índice del ítems que contiene la cadena. Devuelve -1 si la cadena no es encontrada.
"GetItemCount"	Devuelve el número de ítems listados.
"GetSelected" [, opciones]	Devuelve una cadena conteniendo el índice de ítem de los ítems seleccionados. Si opciones=0 (por defecto) solamente el primer ítem seleccionado es devuelto. Si opciones=1 entonces todos los ítems seleccionados son devueltos delimitados por , ejemplo: "0 3 4 10". Si no existen ítems seleccionados una cadena vacía "" es devuelta.
"GetSelectedCount"	Devuelve el número de ítems que son seleccionados.
"GetSubItemCount"	Devuelve el número de subítems.
"GetText", Item, SubItem	Devuelve el texto de un ítem/subítem.
"IsSelected", Item	Devuelve 1 si el ítem es seleccionado, de otro modo devuelve 0.
"Select", From [, To]	Selecciona uno o más ítems.
"SelectAll"	Selecciona todos los ítems.

"SelectClear"	Limpia la lista de ítems seleccionados.
"SelectInvert"	Invierte la selección actual.
"ViewChange", "view"	Cambia la vista actual. Las vistas válidas son "list", "details", "smallicons", "largeicons".

Todos los ítems/subítems son de índice de base 0. Esto significa que el primer ítem/subitem en una lista es 0, el segundo es 1, y así sucesivamente.

En una vista "Details" de un control ListView32, el "ítem" puede ser tomado como la "fila" y el "subitem" como la "columna".

Comentarios

Algunos comandos pueden fallar cuando usamos un proceso de AutoIt de 32 bit para leer desde un proceso de 64 bit. De la misma forma los comandos pueden fallar si utilizamos procesos de AutoIt de 64-bit para leer desde procesos de 32-bit.

Relativo

[ControlCommand](#)

Ejemplo

```
ControlListView(@ProgramFilesDir & "\AutoIt3", "", "SysListView321", "SelectAll")
ControlListView(@ProgramFilesDir & "\AutoIt3", "", "SysListView321", "Deselect", 2, 5)
MsgBox(0, "", ControlListView(@ProgramFilesDir & "\AutoIt3", "", "SysListView321", "GetText",
9, 0) )
MsgBox(0, "", ControlListView(@ProgramFilesDir & "\AutoIt3", "", "SysListView321",
"FindItem", "14 KB", 1) )
MsgBox(0, "", ControlListView(@ProgramFilesDir & "\AutoIt3", "", "SysListView321",
"GetSelected", 1) )
```

ControlMove

Mueve un control dentro de una ventana.

ControlMove ("título", "texto", ID del control, x, y [, ancho [, altura]])

Parámetros

título	El título de la ventana a mover.
--------	----------------------------------

texto	El texto de la ventana a mover.
ID del control	El control para interactuar. Ver Controles .
x	Coordenadas X a mover, relativo a la ventana cliente.
y	Coordenadas Y a mover, relativo a la ventana cliente.
ancho	[opcional] Nuevo ancho de la ventana.
altura	[opcional] Nuevo alto de la ventana.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana/control no es encontrada.

Comentarios

Si x e y son iguales a [Default](#) no ocurre movimiento, simplemente reajuste.

Relativo

[ControlCommand](#), [ControlGetPos](#)

Ejemplo

`ControlMove("[CLASS:Notepad]", "", "MDIClient1", 0, 0, 200, 200)`

ControlSend

Envía una cadena de caracteres a un control.

`ControlSend ("título", "texto", ID del control, "cadena" [, flag])`

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.

ID del control	El control para interactuar. Ver Controles .
cadena	Cadena de caracteres para enviar al control.
flag	[opcional] Cambia cómo las "teclas" son procesadas: flag = 0 (por defecto), El texto contiene caracteres especiales como + para indicar SHIFT y {LEFT} para indicar flecha izquierda. flag = 1, las teclas son enviadas naturalmente.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana/control no es encontrado.

Comentarios

ControlSend funciona de manera similar a [Send\(\)](#) pero este puede enviar teclas directamente a una ventana/control, en vez de simplemente a la ventana activa.

ControlSend es solamente poco confiable para comandos de prompts que funcionan de forma diferente a las ventanas normales(chequea condiciones físicas de estado más bien que enviar mensaje de teclas). Para ventanas normales ControlSend es mucha más confiable que un simple send() - y este sí envía shift, ctrl, alt etc.

Como mención en la ayuda de Send el teclado que envía diferentes caracteres cuando en CAPS LOCK y usando el Shift Key no puede ser simulada. Un ejemplo es el Teclado Czech. Un buen trabajo es utilizar ControlSetText.

El control puede necesitar primeramente tomar el foco con el comando ControlFocus, especialmente cuando se refiere al ID del control creado por el script mismo.

Opt("SendKeyDelay", ...) altera la duración de la pausa entre los envíos de teclazos. Opt("SendKeyDownDelay", ...) altera la longitud del tiempo en que una tecla es presionada antes de ser liberada.

Relativo

[ControlCommand](#), [Send](#), [ControlSetText](#), [ControlFocus](#), [SendKeyDelay \(Option\)](#),
[SendKeyDownDelay \(Option\)](#)

Ejemplo

```
ControlSend("[CLASS:Notepad]", "", "Edit1", "This is a line of text in the notepad window")
```

ControlSetText

Establece texto de un control.

ControlSetText ("título", "texto", ID del control, "nuevo texto" [, flag])

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .
nuevo texto	El nuevo texto para ser establecido en el control.
flag	[opcional] cuando es diferente de 0 (por defecto) forzará a la ventana a redibujarse (actualizarse).

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana/control no es encontrada.

Relativo

[ControlGetText](#), [ControlCommand](#), [ControlSend](#), [SplashTextOn](#)

Ejemplo

```
WinWait("[CLASS:Notepad]")
ControlSetText("[CLASS:Notepad]", "", "Edit1", "El nuevo texto aquí")
```

ControlShow

Muestra un control que ha sido ocultado.

ControlShow ("título", "texto", ID del control)

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana/control no es encontrada.

Relativo

[ControlHide](#), [ControlEnable](#), [ControlCommand](#)

Ejemplo

```
ControlShow("[CLASS:Notepad]", "", "MDIClient1")
```

ControlTreeView

Envía un comando a un control TreeView32.

ControlTreeView ("título", "texto", ID del control, "comando" [, opciones1 [, opciones2]])

Parámetros

título	El título de la ventana a acceder.
texto	El texto de la ventana a acceder.
ID del control	El control para interactuar. Ver Controles .
comando	El comando para enviar al control (ver más abajo).
opciones1	[opcional] Parámetros adicionales requeridos para algunos comandos.
opciones2	[opcional] Parámetros adicionales requeridos para algunos comandos.

Valor de Retorno

Depende del comando mostrado en la tabla más abajo. En caso de un error (como un comando inválido o la ventana/control no encontrada) entonces @error es fijado a 1.

Comando, Opción1, Opción2	Operación
"Check", "ítem"	Chequea un ítem (si el ítem lo soporta).
"Collapse", "ítem"	Colapsa un ítem para ocultar sus hijos.
"Exists", "ítem"	Devuelve 1 si un ítem existe, de otro modo 0.
"Expand", "ítem"	Expande un ítem para mostrar sus hijos.
"GetItemCount", "ítem"	Devuelve el número de hijos de un ítem seleccionado.
"GetSelected" [, UseIndex]	Devuelve la referencia del ítem de una selección actual usando la referencia del texto del ítem (o la referencia del índice si UseIndex es fijado a 1).
"GetText", "ítem"	Devuelve el texto de un ítem.
"IsChecked"	Devuelve el estado de un ítem. 1:chequeado, 0:des-chequeado, -1:no es un checkbox (caja de chequeo).
"Select", "ítem"	Selecciona un ítem.
"Uncheck", "ítem"	Des-chequea un ítem (si el ítem lo soporta).

El parámetro "ítem" es un parámetro de cadena que es usado para referenciar un ítem particular en un treeview usando una combinación de texto e índice. Los índices son de base 0. Por ejemplo:

```

Encabezado1
----> Enca1SubItem1
----> Enca1SubItem2
----> Enca1SubItem3
----> ----> Enca1S1SubItem1
Heading2
Heading3

```

Cada "nivel" es separado por un carácter |. Un índice es precedido por el carácter #. Ejemplos:

Ítem	Referencia del Item
Encabezado2	"Encabezado2" o "#1"
Enca1SubItem2	"Encabezado1 Enca1SubItem2" o "#0 #1"
Enca1S1SubItem1	"Encabezado1 Enca1SubItem3 Enca1S1SubItem1" o "#0 #2 #0"

Las referencias pueden estar mezcladas "Encabezado1|#1".

Comentarios

Como Autolt es una aplicación de 32-bit algunos comandos no están disponibles cuando referencia a aplicaciones de 64-bit; un ejemplo es el Explorador bajo Windows 64-bit.

Relativo

[ControlCommand](#)

Ejemplo

; Author: Zedna

```
#include <GUIContstantsEx.au3>
#include <TreeviewConstants.au3>
#include <WindowsConstants.au3>

$gui = GUICreate("ControlTreeview test", 212, 212)
$treeview = GUICtrlCreateTreeView(6, 6, 200, 160, BitOR($TVS_HASBUTTONS, $TVS_HASLINES,
$TVS_LINESATROOT, $TVS_CHECKBOXES), $WS_EX_CLIENTEDGE)
$h_tree = ControlGetHandle($gui, "", $treeview)

$root = GUICtrlCreateTreeViewItem("Root", $treeview)
$item1 = GUICtrlCreateTreeViewItem("Item 1", $root)
$item2 = GUICtrlCreateTreeViewItem("Item 2", $root)
$item3 = GUICtrlCreateTreeViewItem("Item 3", $root)
$item4 = GUICtrlCreateTreeViewItem("Item 4", $root)
$item41 = GUICtrlCreateTreeViewItem("Item 41", $item4)
$item42 = GUICtrlCreateTreeViewItem("Item 42", $item4)
$item5 = GUICtrlCreateTreeViewItem("Item 5", $root)

GUISetState(@SW_SHOW)

; algunos ejemplos
ControlTreeView ($gui, "", $h_tree, "Expand", "Root")
```

```

ControlTreeView ($gui, "", $h_tree, "Existe", "Root/Item 4")
ControlTreeView ($gui, "", $h_tree, "Checkeado", "Root/Item 4")
ControlTreeView ($gui, "", $h_tree, "Seleccionado", "Root/Item 4")
ControlTreeView ($gui, "", $h_tree, "Expandido", "Root/Item 4")

```

```

While 1
  $msg = GUIGetMsg()
  Select
    Case $msg = $GUI_EVENT_CLOSE
      ExitLoop
    EndSelect
  WEnd

```

StatusbarGetText

Recupera el texto de una barra de estado estándar.

StatusbarGetText ("título" [, "texto" [, parte]])

Parámetros

título	El título de la ventana a revisar.
texto	[opcional] El texto de la ventana a revisar.
parte	[opcional] El número de "parte" de la barra de estado a leer - el predefinido es 1. 1 es la primer posible parte y usualmente la única que contenga mensaje útiles como "Listo" "Cargando...", etc.

Valor de Retorno

Con Éxito: Devuelve el texto leído.

Al Fallar: Devuelve "" (cadena vacía) y establece el valor de @error en 1 si no hay texto que pueda ser leído.

Comentarios

Esta función intenta leer en una barra estándar de estado en una ventana (Control común de Microsoft: msctls_statusbar32) Algunos programas usan su propias barras de estado o versiones especiales del control común de MS que StatusbarGetText no puede leer. Por ejemplo, StatusbarText no trabaja en el programa TextPad; sin embargo, en la primera región de la barra de estado de TextPad puede ser leída usando ControlGetText("TextPad", "", "HSStatusBar1").

StatusbarGetText puede trabajar en ventanas que están minimizadas incluso ocultas.

Relativo

[ControlGetText](#), [ControlCommand](#)

Ejemplo

```
AutoItSetOption("WinTitleMatchMode", 2)
```

```
$x = StatusbarGetText("Internet Explorer")
MsgBox(0, "Barra de estado de Internet Explorer dice:", $x)
```

WinActivate

Activa (toma el foco) una ventana.

WinActivate ("título" [, "texto"])

Parámetros

título	El título de la ventana a activar. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana a activar.

Valor de Retorno

Con Éxitos Devuelve el identificador (handle) de la ventana.

Al Fallar Devuelve 0 si no se encuentra la ventana o no puede ser activada.

Comentarios

Usted puede utilizar la función WinActive para revisar si WinActivate ha sido satisfactoria. Si múltiples ventanas coinciden con el criterio, la ventana activa más reciente es la activada. WinActivate funciona en ventanas minimizadas. Sin embargo, una ventana que es "Siempre visible" podría todavía permitir una ventana activada.

@extended contiene información adicional acerca de cómo está el proceso de la ventana.

Relativo

[WinClose](#), [WinsetState](#), [WinTitleMatchMode \(Option\)](#), [WinKill](#), [WinMove](#)

Ejemplo

```
WinActivate("[CLASS:Notepad]", "")
```

WinActive

Chequea si una ventana está abierta y ha sido actualmente activada.

WinActive ("título" [, "texto"])

Parámetros

título	El título de la ventana a revisar. Ver Para definición especial de título.
texto	[opcional] El texto de la ventana a revisar.

Valor de Retorno

Con Éxitos Devuelve el identificador (handle) de la ventana si esta activa.

Al Fallar Devuelve 0 de otro modo.

Relativo

[WinExists](#), [WinWait](#), [WinWaitActive](#), [WinWaitClose](#), [WinWaitNotActive](#), [WinTitleMatchMode \(Option\)](#)

Ejemplo

```
If WinActive("[CLASS:Notepad]") Then  
    MsgBox(0, "", "La ventana está activa")  
EndIf
```

WinClose

Cierra una ventana

WinClose ("título" [, "texto"])

Parámetros

título	El título de la ventana a cerrar. Ver Para definición especial de título.
--------	---

texto	[opcional] El texto de la ventana a cerrar.
-------	--

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana no es encontrada.

Comentarios

Esta función envía un mensaje de cierre a una ventana, el resultado depende de la ventana (esta puede preguntar si salvar datos, etc.), para forzar a una ventana a cerrarse, use la función WinKill. Si múltiples ventanas coinciden con el criterio, la ventana activa más reciente es la cerrada.

Relativo

[WinActivate](#), [WinExists](#), [WinKill](#), [WinSetState](#), [WinTitleMatchMode \(Option\)](#), [ProcessClose](#), [WinMove](#)

Ejemplo

`WinClose("[CLASS:Notepad]", "")`

WinDetectHiddenText

Especifica si el texto oculto de la ventana puede ser "visto" para funciones de coincidencia en ventana.

0 = No detecta texto oculto (por defecto).

1 = Detecta texto oculto.

WinExists

Chequea si una ventana especificada existe.

`WinExists ("título" [, "texto"])`

Parámetros

título	El título de la ventana a revisar. Ver Para definición especial de título .
--------	---

texto	[opcional] El texto de la ventana a revisar.
-------	---

Valor de Retorno

Con Éxitos Devuelve 1 si la ventana existe.

Al Fallar Devuelve 0.

Comentarios

WinExists devolverá 1 aun cuando la ventana esta oculta.

Relativo

[WinActive](#), [WinWait](#), [WinWaitActive](#), [WinWaitClose](#), [WinWaitNotActive](#), [WinTitleMatchMode \(Option\)](#), [ProcessExists](#), [WinClose](#)

Ejemplo

```
If WinExists("[CLASS:Notepad]") Then
    MsgBox(0, "", "La ventana existe")
Endif
```

WinFlash

Provoca parpadeo de una ventana en la barra de tareas.

WinFlash ("título" [, "texto" [, parpadeo [, espera]]])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana a leer.
parpadeo	[opcional] cantidad de veces que parpadea la ventana. Por defecto es 4.
espera	[opcional] El tiempo en milisegundos de espera entre cada parpadeo. Por defecto es 500 ms.

Valor de Retorno

Ninguno

Comentarios

Esta función efectiva para captar la atención del usuario!

Relativo

[WinSetState](#)

Ejemplo

; Parpadea la ventana 4 veces con una pausa de medio segundo
WinFlash("[CLASS:Notepad]","", 4, 500)

WinGetCaretPos

Devuelve las coordenadas de intercalamiento para la ventana en primer plano.

WinGetCaretPos ()

Parámetros

Ninguno

Valor de Retorno

Con Éxitos Devuelve un arreglo de 2 dimensiones conteniendo la siguientes información:

\$arreglo[0] = X coordenada

\$arreglo[1] = Y coordenada

Al Fallar Establece @error a 1.

Comentarios

WinGetCaretPos puede no devolver un valor exacto para aplicaciones con Interfaces de Múltiples Documentos(MDI) si el modo CaretCoordMode es utilizado. Ver ejemplo para una demostración. Nota: Algunas aplicaciones reportan coordenadas estáticas indistintamente de su posición de intercalamiento.

Relativo

[CaretCoordMode \(Option\)](#)

Ejemplo

```
$a = WinGetCaretPos()  
If Not @error Then ToolTip("Primer método de posicionamiento", $a[0], $a[1])  
sleep(2000)
```

```
$b = _CaretPos()  
If Not @error Then ToolTip("Segundo método de posicionamiento", $b[0], $b[1])  
sleep(2000)
```

```

; Método más confiable para obtener las coordenadas del carro en MDI de editores de texto..
Func _CaretPos()
Local $x_adjust = 5
Local $y_adjust = 40

Opt("CaretCoordMode", 0)           ;modo relativo
Local $c = WinGetCaretPos()         ;coordenadas del carro relativas
Local $w = WinGetPos("")            ;coordenadas de ventana
Local $f = ControlGetFocus("", "")  ;región del texto "handle"
Local $e = ControlGetPos("", "", $f) ;coordenadas de la región del texto

Local $t[2]
If IsArray($c) and IsArray($w) and IsArray($e) Then
    $t[0] = $c[0] + $w[0] + $e[0] + $x_adjust
    $t[1] = $c[1] + $w[1] + $e[1] + $y_adjust
    Return $t   ;coordenadas absolutas de la pantalla para el mouse
Else
    SetError(1)
EndIf
EndFunc

```

WinGetClassList

Captura las clases internas de una ventana.

WinGetClassList ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana a leer.

Valor de Retorno

Con Éxitos Devuelve una cadena conteniendo la clase de ventana leída.

Al Fallar Devuelve "" y fija @error a 1 si la ventana no coincide con el criterio.

Comentarios

Los nombres de Clases son separados por caracteres de retorno de línea (@LF). La función WinGetClassList funciona en ventanas minimizadas y ocultas. Textos de hasta 64 KB pueden ser devueltos. Si múltiples ventanas coinciden con el criterio, Las clases son leídas desde la ventana activa más reciente.

Relativo

[WinGetText](#), [ControlCommand](#)

Ejemplo

```
$text = WinGetClassList("[CLASS:Notepad]", "")  
MsgBox(0, "Texto leído:", $text)
```

WinGetClientSize

Captura el tamaño de una ventana determinada de área cliente.

WinGetClientSize ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana a leer.

Valor de Retorno

Con Éxitos Devuelve un arreglo de 2 dimensiones conteniendo la siguientes información:

\$array[0] = Ancho de la ventana de área cliente

\$array[1] = Altura de la ventana de área cliente

Al Fallar Devuelve 0 y fija @error un 1 si la ventana no es encontrada.

Comentarios

Si la ventana es minimizada, el ancho y el alto devuelto es cero. Sin embargo, WinGetClientSize funciona correctamente en ventanas ocultas (no minimizada). Si la ventana tiene el título "Program Manager", la función devolverá el tamaño del escritorio. Si múltiple ventanas coinciden con el criterio, la ventana activa más reciente es usada.

Relativo

[WinGetPos](#), [WinMove](#)

Ejemplo

```
$size = WinGetClientSize("[active]")  
MsgBox(0, "El tamaño de la ventana hija activa es (ancho,alto):", $size[0] & " " & $size[1])
```

WinGetHandle

Devuelve el identificador interno de una ventana.

WinGetHandle ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana a leer.

Valor de Retorno

Con Éxitos Devuelve el identificador interno de una ventana.

Al Fallar Devuelve "" (cadena vacía) y establece @error a 1 si la ventana no coincide con el criterio.

Comentarios

Esta función le permite a usted usar identificadores (handles) para especificar ventanas más que solamente el "título" y "texto".

La ventaja es que si usted obtiene el identificador puede acceder a la ventana aún cuando su título haya cambiado.

Relativo

[WinSetTitle](#), [GUICreate](#), [WinList](#)

Ejemplo

; Identifica la ventana del notepad que contiene el texto "esta" y obtiene el identificador (handle) de la misma

; Cambia el WinTitleMatchMode que ahora soporta classnames y handles (identificadores)
AutoItSetOption("WinTitleMatchMode", 4)

; Obtiene el identificador (handle) de la ventana del notepad que contiene "this one"

\$Handle = WinGetHandle("classname=Notepad", "esta")

If @error Then

 MsgBox(4096, "Error", "No se encuentra la ventana correcta")

Else

 ; Envía algunos textos directamente al control Edit de la ventana

```
ControlSend($Handle, "", "Edit1", "AbCdE")
EndIf
```

WinGetPos

Devuelve la posición y tamaño de una ventana dada.

WinGetPos ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título.
texto	[opcional] El texto de la ventana a leer.

Valor de Retorno

Con Éxitos Devuelve un arreglo de 4 elementos conteniendo las siguientes informaciones:

\$arreglo[0] = Posición X

\$arreglo[1] = Posición Y

\$arreglo[2] = Ancho

\$arreglo[3] = Alto

Al Fallar Devuelve 0 y fija @error a 1 si la ventana no es encontrada.

Comentarios

WinGetPos devuelve números negativos como -32000 para ventanas minimizadas, pero funciona bien con ventanas ocultas (no minimizadas).

Si es usado el título de ventana "Program Manager", la función devolverá el tamaño del escritorio. Si múltiples ventanas coinciden con el criterio, la ventana activa más reciente es usada.

Relativo

[WinGetClientSize](#), [WinMove](#), [WinGetState](#)

Ejemplo

```
$size = WinGetPos("[active]")
```

```
MsgBox(0, "estado de la ventana activa (x,y,ancho,alto):", $size[0] & " " & $size[1] & " " &  
$size[2] & " " & $size[3])
```

WinGetProcess

Devuelve el ID del proceso (PID) asociado a una ventana.

WinGetProcess ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana para leer.

Valor de Retorno

Con Éxitos Devuelve el número ID del proceso (PID).

Al Fallar Devuelve -1.

Relativo

[ProcessWait](#), [ProcessWaitClose](#), [ProcessList](#)

Ejemplo

```
Run("notepad.exe")  
$pid = WinGetProcess("[CLASS:Notepad]")  
MsgBox(4096, "PID es", $pid)
```

WinGetState

Devuelve el estado de una ventana dada.

WinGetState ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana para leer.

Valor de Retorno

Con Éxitos Devuelve un valor indicando el estado de la ventana. Múltiples valores son adicionados, use BitAND() para examinar la parte que le interesa:

1 = Ventana existe

2 = Ventana es visible

4 = Ventana está habilitada

8 = Ventana activa

16 = Ventana minimizada

32 = Ventana maximizada

Al Fallar Devuelve 0 y establece @error a 1 si la ventana no es encontrada.

Relativo

[BitAND](#), [WinGetPos](#)

Ejemplo

```
; Revisa si una nueva ventana del Notepad es minimizada  
$state = WinGetState("[CLASS:Notepad]", "")
```

```
; Si es "minimizada" entonces?  
If BitAnd($state, 16) Then  
    MsgBox(0, "Example", "Ventana minimizada")  
EndIf
```

WinGetText

Devuelve el texto de una ventana.

WinGetText ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana para leer.

Valor de Retorno

Con Éxitos Devuelve una cadena conteniendo el texto de la ventana leída.

Al Fallar Devuelve 0 si el título no coincide.

Comentarios

Hasta 64KB de texto pueden ser devueltos. WinGetText funciona en ventanas minimizadas, pero solo puede funcionar en ventanas ocultas si AutoItSetOption("WinDetectHiddenText", 1) Si múltiples ventanas coinciden con el criterio para WinGetText, la información de la ventana más recientemente abierta es devuelto.

Use WinGetText("[active]") el texto de la ventana activa.

Relativo

[WinGetTitle](#), [ControlGetText](#), [WinGetClassList](#)

Ejemplo

```
$text = WinGetText("[CLASS:Notepad]", "")  
MsgBox(0, "Texto leído:", $text)
```

WinGetTitle

Devuelve el título completo de la ventana.

WinGetTitle ("título" [, "texto"])

Parámetros

título	El título de la ventana para leer. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana para leer.

Valor de Retorno

Con Éxitos Devuelve una cadena conteniendo el título completo de la ventana.

Al Fallar Devuelve el número 0 si el título no coincide.

Comentarios

`WinGetTitle("[active]")` devuelve el título de la ventana activa. WinGetTitle funciona en ventanas minimizadas y ocultas. Si múltiples ventanas coinciden con el criterio, la ventana más recientemente activa es usada.

Relativo

[AutoItWinGetTitle](#), [WinGetText](#), [WinSetTitle](#), [WinTitleMatchMode \(Option\)](#)

Ejemplo

```
$title = WinGetTitle("[CLASS:Notepad]", "")  
MsgBox(0, "Título completo:", $title)
```

WinKill

Fuerza una ventana a cerrar.

WinKill ("título" [, "texto"])

Parámetros

título	El título de la ventana a cerrar. Ver Para definición especial de título .
texto	[opcional] El texto de la ventana para cerrar.

Valor de Retorno

Ninguno. (Siempre devuelve 1 indistintamente de lo que ocurra)

Comentarios

La diferencia entre este función y WinClose es que WinKill termina forzadamente la ventana si esta no se cierra. Consecuentemente, un usuario puede no tener tiempo para interactuar con la posible ventana de diálogo de guardar los datos. Aunque WinKill puede trabajar en ventanas minimizadas y ocultas, algunas ventanas (Explorador de Windows) sólo pueden ser terminadas utilizando WinClose.

Relativo

[WinActivate](#), [WinClose](#), [WinSetState](#), [ProcessClose](#)

Ejemplo

```
WinKill("[CLASS:Notepad] ", "")
```

WinList

Devuelve una lista de ventanas.

WinList ("título" [, "texto"])

Parámetros

título	[opcional] El título de la ventana para leer.
texto	[opcional] El texto de la ventana para leer.

Valor de Retorno

Devuelve un arreglo de ventanas coincidentes e identificadores.

Comentarios

Si el título y el texto no son devueltos entonces se devuelven todas las ventanas de nivel superior.

El arreglo devuelto es bidimensional y contiene lo siguientes elementos:

\$arreglo[0][0] = Número de ventanas devueltas

\$arreglo[1][0] = Título de la primera ventana

\$arreglo[1][1] = Identificador de la primera ventana(HWND)

\$arreglo[2][0] = Título de la segunda ventana

\$arreglo[2][1] = Identificador de la segunda ventana(HWND)

...

\$arreglo[n][0] = Título de la enésima ventana

\$arreglo[n][1] = Identificador de la enésima ventana(HWND)

Relativo

[WinGetHandle](#)

Ejemplo

`$var = WinList()`

```

For $i = 1 to $var[0][0]
; Solo muestra las ventanas visibles que tienen títulos
If $var[$i][0] <> "" AND IsVisible($var[$i][1]) Then
    MsgBox(0, "Detalles", "Títulos=" & $var[$i][0] & @LF & "Identificador=" & $var[$i][1])
EndIf
Next

Func IsVisible($handle)
If BitAnd( WinGetState($handle), 2 ) Then
    Return 1
Else
    Return 0
EndIf
EndFunc

```

WinMenuItem

Invoca el ítem de un menú dentro de una ventana.

```
WinMenuSelectItem ("título", "texto", "item" [, "item" ]]]]]]])
```

Parámetros

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el menu no es encontrado.

Comentarios

Usted debe notar que los item de menú que son guiones bajo () actualmente contienen el carácter & para indicar guión bajo. De esta forma, el menú item **File** requerirá tener el texto "&File", y **Convert** requerirá "Con&vert" Usted puede acceder a un menú item de nivel 6 como máximo; y la ventana puede ser inactiva, minimizada, y/o oculta.

WinMenuItem trabaja solamente con menús estándares. Desafortunadamente, muchos menus todavía son actualmente escritos o "simulan" barras de herramientas para ser menús. Esta función es certera para la mayoría de aplicaciones directamente de Microsoft.

Relativo

[ControlCommand](#), [Send](#)

Ejemplo

```
; Esto seleccionará File, Page Setup en notepad  
WinMenuItem("[CLASS:Notepad]","", "&File", "Page Set&up...")
```

WinMinimizeAll

Minimiza todas las ventanas.

WinMinimizeAll ()

Parámetros

Ninguno.

Valor de Retorno

Ninguno.

Comentarios

Send("#m") es una posible alternativa.

Relativo

[WinMinimizeAllUndo](#), [WinSetState](#)

Ejemplo

```
WinMinimizeAll()
```

WinMinimizeAllUndo

Deshace la operación de la función WinMinimizeAll.

WinMinimizeAllUndo ()

Parámetros

Ninguno

Valor de Retorno

Ninguno

Comentarios

Send("#+m") es una posible alternativa.

Relativo

[WinMinimizeAll](#), [WinSetState](#)

Ejemplo

WinMinimizeAllUndo()

WinMove

Mueve y/o cambia el tamaño de una ventana.

WinMove ("título", "texto", x, y [, ancho [, altura[, velocidad]]])

Parámetros

título	El título de la ventana a mover/redimensionar. Ver Definición de Título especial .
texto	El texto de la ventana a mover/redimensionar.
x	Movimiento en las coordenadas X.
y	Movimiento en las coordenadas Y.
ancho	[opcional] Nuevo ancho de la ventana.

altura	[opcional] Nuevo alto de la ventana.
velocidad	[opcional] la velocidad al mover la ventana en el rango de 1 (rápido) a 100 (lento). Si no es definido se moverá al instante.

Valor de Retorno

Con Éxitos Devuelve el identificador (handle) de la ventana.

Al Fallar Devuelve 0 si la ventana no es encontrada.

Comentarios

WinMove no tiene efecto en ventanas minimizadas, pero WinMove funciona en ventanas ocultas.

Si el alto o el ancho son pequeños (o negativos), entonces la ventana tendrá como mínimo 112 x 27 pixeles. Si el alto o el ancho son grandes, la ventana no sobrepasará los límites de [12+@DesktopWidth] x [12+@DesktopHeight] pixeles.

Valores negativos son permitidos para las coordenadas X y Y. En efecto, usted puede mover una ventana fuera de la pantalla; y si el programa de la ventana recuerda la última posición, la ventana aparecerá ubicada en la esquina (pero completamente en pantalla) la próxima vez que usted ejecute el programa.

Si múltiples ventanas coinciden con el criterio, la ventana más recientemente activa será tomada.

Si los valores de X y Y son iguales a [Palabra clave por defecto](#) no ocurre movimiento, simplemente cambia de tamaño.

Cuando se usa la velocidad el movimiento es dado con el nuevo tamaño.

Relativo

[WinActivate](#), [WinClose](#), [WinGetClientSize](#), [WinGetPos](#), [WinSetState](#)

Ejemplo

```
WinMove("[CLASS:Notepad]", "", 0, 0, 200, 200)
```

WinSearchChildren

Permite las rutinas de búsqueda de ventana, busque ventanas dependientes (ventana hija) como ventanas de nivel sobresaliente (top-level).

0 = Sólo busca las ventanas de nivel mayor (por defecto)

1 = Busca en todas (nivel mayor y dependientes)

WinSetOnTop

Cambia una ventana colocándole el atributo "Siempre Visible" (Always On Top)

WinSetOnTop ("título", "texto", bandera)

Parámetros

título	El título de la ventana a afectar. Ver Para definición especial de título
texto	El texto de la ventana por afectar.
bandera	Determina si la ventana debe tener el valor de "Siempre visible" ("TOPMOST") establecido. 1= establecer una bandera superior, 0 = remover el valor anterior.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana no es encontrada.

Comentarios

Los programas que adicionan un menú contextual con la opción "Always On Top" no pueden actualizar esta entrada del menú para reflejarlo, Autolt- induce el cambio al estado TOPMOST

Relativo

[WinSetState](#)

Ejemplo

`WinSetOnTop("[CLASS:Notepad]", "", 1)`

WinSetState

Muestra, Oculta, minimiza, maximiza, o restaura una ventana.

WinSetState ("título", "texto", bandera)

Parámetros

título	El título de la ventana a mostrar. Ver Para definición especial de título
texto	El texto de la ventana para mostrar.
bandera	El modo en que se "mostrará" el programa una vez ejecutado: @SW_HIDE = Ventana oculta @SW_SHOW = Muestra una ventana previamente oculta @SW_MINIMIZE = Ventana minimizada @SW_MAXIMIZE = Ventana maximizada @SW_RESTORE = Deshace una operación de maximización o minimización @SW_DISABLE = Deshabilita la ventana @SW_ENABLE = Habilita la ventana

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana no es encontrada.

Comentarios

Si múltiples ventanas coinciden con el criterio, la más recientemente activa es usada.

@SW_MINIMIZE y @SW_MAXIMIZE siempre trabajan en ventanas de diálogo modal.

Relativo

[WinActivate](#), [WinClose](#), [ControlHide](#), [WinFlash](#), [WinKill](#), [WinMinimizeAll](#), [WinMinimizeAllUndo](#), [WinMove](#), [WinSetOnTop](#)

Ejemplo

```
WinSetTitle("[CLASS:Notepad]", "", @SW_HIDE)
Sleep(3000)
WinSetTitle("[CLASS:Notepad]", "", @SW_SHOW)
```

WinSetTitle

Cambia el título de una ventana.

WinSetTitle ("título", "texto", "Nuevotítulo")

Parámetros

título	El título de la ventana a cambiar. Ver Para definición especial de título
texto	El texto de la ventana a cambiar.
Nuevotítulo	El nuevo título que tendrá la ventana.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si la ventana no es encontrada.

Comentarios

Si múltiples ventanas coinciden con el criterio la más recientemente activa es usada.

Relativo

[AutoltWinSetTitle](#), [WinGetTitle](#), [WinTitleMatchMode \(Option\)](#), [WinGetHandle](#)

Ejemplo

```
WinSetTitle("[CLASS:Notepad] ", "", "My New Notepad")
```

WinTextMatchMode

Altera el método que es usado para coincidir con el texto de la ventana durante las operaciones de búsqueda de ventana.

1 = Completo / modo lento (por defecto)

2 = Modo rápido

En modo rápido, Autolt puede usualmente sólo "ver" en el texto de diálogo, texto en botón y títulos de algunos controles. En el modo por defecto mucho más texto puede ser visto (por instancia el contenido de la ventana de Notepad).

Si tiene problemas de rendimiento cuando realiza muchas búsquedas de ventana, puede que ayude utilizar el modo rápido.

WinSetTrans

Establece el nivel de trasparencia de una ventana. (Windows 2000/XP o superior)

WinSetTrans ("título", "texto", transparencia)

Parámetros

título	El título de la ventana a cambiar. Ver Para definición especial de título
texto	El texto de la ventana a cambiar.
transparencia	Un número en el rango de 0 - 255. Para el menor número, la mayor transparencia. 255 = Solido, 0 = Invisible.

Valor de Retorno

Con Éxitos Devuelve valor diferente de cero.

Al Fallar Devuelve 0, @error será establecido a 1 si la función no es soportada en el SO.

Comentarios

Requiere Windows 2000/XP o superior. La configuración del color de la pantalla debe ser mayor o igual a 16-bit.

Relativo

Ninguno.

Ejemplo

```
Opt("WinTitleMatchMode", 2); Subcadenas coincidentes  
WinSetTrans("Notepad", "", 170); Hacer la ventana del Notepad semi transparente
```

WinTitleMatchMode

Altera el método que es usado para coincidir con los títulos de las ventanas.

1 = Coincide con el título desde el inicio (por defecto)

2 = Coincide cualquier cadena en el título

3 = Coincidencia exacta de título

4 = Modo avanzado, ver [Títulos de ventanas y texto \(Avanzados\)](#)

-1 a -4 = fuerza coincidencia en minúsculas de acuerdo con otro tipo de coincidencia.

WinWait

Pausa la ejecución del script hasta que la ventana solicitada exista.

WinWait ("título" [, "texto" [, Tiempo espera]])

Parámetros

título	El título de la ventana a revisar. Ver Para definición especial de título
texto	[opcional] El texto de la ventana a revisar.
Tiempo espera	[opcional] Tiempo de espera en segundos.

Valor de Retorno

Con Éxitos Devuelve el identificador (handle) de la ventana solicitada.

Al Fallar Devuelve 0 si el tiempo límite es alcanzado.

Relativo

[WinActive](#), [WinExists](#), [WinWaitActive](#), [WinWaitClose](#), [WinWaitNotActive](#), [WinWaitDelay](#), [\(Option\)](#), [ProcessWait](#)

Ejemplo

;Espera por una ventana "[CLASS:Notepad]" para salir

```
Run("notepad")
WinWait("[CLASS:Notepad]")
```

;Espera un máximo de 5 segundos "[CLASS:Notepad]" para salir

```
WinWait("[CLASS:Notepad]", "", 5)
```

WinWaitActive

Pausa la ejecución de un script hasta que la ventana solicitada se activa.

WinWaitActive ("título" [, "texto" [, Tiempo espera]])

Parámetros

título	El título de la ventana a revisar. Ver Para definición especial de título
texto	[opcional] El texto de la ventana a revisar.
Tiempo espera	[opcional] Tiempo espera en segundos.

Valor de Retorno

Con Éxitos Devuelve el identificador (handle) de la ventana solicitada.

Al Fallar Devuelve 0 si el tiempo límite es alcanzado.

Relativo

[WinActive](#), [WinExists](#), [WinWait](#), [WinWaitClose](#), [WinWaitNotActive](#), [WinWaitDelay \(Option\)](#), [ProcessWait](#)

Ejemplo

*;Espera por la ventana "[CLASS:Notepad]" para existir y estar activa
WinWaitActive("[CLASS:Notepad]")*

*;Espera un máximo de 5 segundos por "[CLASS:Notepad]" para existir y estar activa
WinWaitActive("[CLASS:Notepad]", "", 5)*

WinWaitClose

Pausa la ejecución del script hasta que la ventana solicitada no exista.

WinWaitClose ("título" [, "texto" [, Tiempo espera]])

Parámetros

título	El título de la ventana a revisar. Ver Definición de título especial
texto	[opcional] El texto de la ventana a revisar.
Tiempo espera	[opcional] Tiempo espera en segundos.

Valor de Retorno

Con Éxitos Devuelve 1.

Al Fallar Devuelve 0 si el tiempo límite es alcanzado.

Relativo

[WinActive](#), [WinExists](#), [WinWait](#), [WinWaitActive](#), [WinWaitNotActive](#), [WinWaitDelay \(Option\)](#), [ProcessWaitClose](#)

Ejemplo

;Wait for the window "Untitled" to not exist

`WinWaitClose("Untitled")`

;Wait a maximum of 5 seconds for "Untitled" to not exist

`WinWaitClose("Untitled", "", 5)`

WinWaitDelay

Altera cómo un script debería hacer una breve pausa después de una operación exitosa relacionada con ventanas. Tiempo en milisegundos a pausar (por defecto=250).

WinWaitNotActive

Pausa la ejecución de un scripts hasta que la ventana especificada no esté activa.

`WinWaitNotActive ("título" [, "texto" [, tiempo de expiración]])`

Parámetros

título	El título de la ventana a revisar. Ver Definición de título especial
texto	[opcional] El texto de la ventana a revisar.
tiempo final	[opcional] Tiempo de expiración en segundos.

Valor de Retorno

Con Éxitos: Devuelve 1.

Al Fallar: Devuelve 0 si ya transcurrió el tiempo límite para esperar el texto.

Comentarios

El script busca o compara una ventana coincidente cada 250 milisegundos o algo así.

Relativo

[WinActive](#), [WinExists](#), [WinWait](#), [WinWaitActive](#), [WinWaitClose](#), [WinWaitDelay \(Option\)](#)

Ejemplo

;Wait for the window "Untitled" to not be active

```
WinWaitNotActive("Untitled")
```

;Wait a maximum of 5 seconds for "Untitled" to not be active

```
WinWaitNotActive("Untitled", "", 5)
```

APÉNDICE

Limitaciones/valores por defectos en AutoIt3

Límites AutoIt3	Valores	Descripción
MAX_LINESIZE	4095	Tamaño máximo de una línea de script
STRBUFFER	4095	Tamaño del buffer de cadena
MAX_ENVSIZE	32767	Tamaño máximo de las variable ENV (entorno)
WINTEXTBUFFER	32767	GetWindowText falla bajo 95 si >65535, WM_GETTEXT fortuitamente falla si > 32767
IDLE	10 ms	Número en ms para espera cuando idling
HOTKEYQUEUESIZE	64	Número máximo de pulsaciones de teclas que pueden ser almacenados
MAXEXECUTERECURSE	384	Número máximo de veces que la función Execute() puede repetirse a sí misma
GUI_MAXWINDOWS	1024	Número máximo de ventanas simultáneas
GUI_MAXCONTROLS	4093	Número máximo de controles en una GUI
GUI_MAXCOLOURS	32	Número máximo de colores en panel cacheados
GUI_MAXPOINTS	256	Número máximo de puntos por página de información gráfica
COM_MAXEVENT	64	Número máximo de eventos COM que pueden ser buffereados
VAR_SUBSCRIPT_MAX	64	Número máximo de dimensiones en un arreglo
MAIN_TIMER_DELAY	750 ms	El tiempo de refrescamiento para la condición de oculto/parpadeo/animación del ícono Tray, es cada 750ms

CMDLINE_MAXPARAMS	63	Número máximo de parámetros
CMDLINEPARAM_MAXLEN	4096	Cada parámetro puede tener algunos caracteres
TRAY_MAXITEMS	505	Número máximo de ítems en el tray menú
TRAY_MAXEVENT	32	Número máximo de eventos que pueden ser almacenados
TRAY_TOOLTIPWIDTH	64	Número máximo de caracteres mostrados en un tray tooltip

valores por defecto	Valores	Descripción
ADLIB delay	250 ms	Espera por defecto en ms entre desencadenadores (triggers) ADLIB. Puede ser definido por AdlibRegister()
		Los siguientes defectos pueden ser cambiados con Opt()
MouseClickDelay	10 ms	Tiempo entre clics del mouse
MouseClickDownDelay	10 ms	Tiempo de clic mantenido
MouseClickDragDelay	250 ms	Espera entre inicio y fin de la operación drag
TCPTimeout	100 ms	Tiempo antes que la función TCP sea abortada

Códigos de Caracteres ASCII

Esta lista resulta útil para las funciones [Asc](#) y [Chr](#). Tomado del [Juego de caracteres ASCII](#)

Caracteres de control (la mayoría son imprimibles; los más usados son resaltados en amarillo.)

Carácter	Dec	Hex	Oct	Descripción
NUL	0	00	000	Carácter Nulo
SOH	1	01	001	Inicio de cabecera, = interruptor de consola

STX	2	02	002	Inicio de texto, modo de mantenimiento en consola HP
ETX	3	03	003	Fin de texto
EOT	4	04	004	Fin de Transmisión, distinto de ETB
ENQ	5	05	005	Solicitud de información, dirigido con ACK; antiguo control flotante de HP
ACK	6	06	006	Confirmación, limpia el carácter ENQ
BEL	7	07	007	Timbre, señal audible... (beep por defecto en Windows)
BS	8	08	010	Retroceso, trabaja en computadora/terminal HP
HT	9	09	011	Tabulación horizontal, mueve hacia el próximo tab stop
LF	10	0a	012	Salto de línea
VT	11	0b	013	Tabulación Vertical
FF	12	0c	014	Salto de forma, Salto de página
CR	13	0d	015	Retorno de carro
SO	14	0e	016	Terminar modo Mayúscula, fijar carácter alternado
SI	15	0f	017	Iniciar modo mayúscula, fijar resumen de carácter por defecto
DLE	16	10	020	Escape del enlace de datos
DC1	17	11	021	XON, con XOFF para pausar listado; ":okay to send"
DC2	18	12	022	Control dispositivo 2, control flotante de modo-bloque
DC3	19	13	023	XOFF, con XON es TERM=18 control flotante
DC4	20	14	024	Control dispositivo 4
NAK	21	15	025	Confirmación negativa
SYN	22	16	026	Sincronización de la comunicación

ETB	23	17	027	Fin de bloque de trasmisión, distinto de EOT
CAN	24	18	030	Línea cancelar, MPE ecos !!!
EM	25	19	031	Fin medio, Interruptor Control-Y
SUB	26	1a	032	Remplazar
ESC	27	1b	033	Escape, el siguiente carácter no se repite
FS	28	1c	034	Separador de archivos
GS	29	1d	035	Separador de grupos
RS	30	1e	036	Separador de registros, terminador de modo-Bloque
US	31	1f	037	Separador de unidades

Caracteres imprimibles (caracteres estándares)

Carácter	Dec	Hex	Oct	Descripción
	32	20	040	Espacio
!	33	21	041	Signo de exclamación
"	34	22	042	Signo de Comillas (" in HTML)
#	35	23	043	Cross hatch (signo de número)
\$	36	24	044	Signo de dolar
%	37	25	045	Signo de porcentaje
&	38	26	046	Ampersand
`	39	27	047	Comilla simple de cierre (apóstrofe)
(40	28	050	Paréntesis de apertura
)	41	29	051	Paréntesis de cierre

*	42	2a	052	Asterisco (star, multiplicación)
+	43	2b	053	Suma
,	44	2c	054	Coma
-	45	2d	055	Guion, dash, menos
.	46	2e	056	Punto
/	47	2f	057	Slant (barra oblicua, división)
0	48	30	060	Cero
1	49	31	061	Uno
2	50	32	062	Dos
3	51	33	063	Tres
4	52	34	064	Cuatro
5	53	35	065	Cinco
6	54	36	066	Seis
7	55	37	067	Siete
8	56	38	070	Ocho
9	57	39	071	Nueve
:	58	3a	072	Doble puntos
;	59	3b	073	Punto y coma
<	60	3c	074	Signo de menor que
=	61	3d	075	Signo de Igual
>	62	3e	076	Signo de mayor que
?	63	3f	077	Signo de interrogación

@	64	40	100	Signo de arroba
A	65	41	101	Mayúscula A
B	66	42	102	Mayúscula B
C	67	43	103	Mayúscula C
D	68	44	104	Mayúscula D
E	69	45	105	Mayúscula E
F	70	46	106	Mayúscula F
G	71	47	107	Mayúscula G
H	72	48	110	Mayúscula H
I	73	49	111	Mayúscula I
J	74	4a	112	Mayúscula J
K	75	4b	113	Mayúscula K
L	76	4c	114	Mayúscula L
M	77	4d	115	Mayúscula M
N	78	4e	116	Mayúscula N
O	79	4f	117	Mayúscula O
P	80	50	120	Mayúscula P
Q	81	51	121	Mayúscula Q
R	82	52	122	Mayúscula R
S	83	53	123	Mayúscula S
T	84	54	124	Mayúscula T
U	85	55	125	Mayúscula U

V	86	56	126	Mayúscula V
W	87	57	127	Mayúscula W
X	88	58	130	Mayúscula X
Y	89	59	131	Mayúscula Y
Z	90	5a	132	Mayúscula Z
[91	5b	133	Llave de cierre
\	92	5c	134	slash inverso (Backslash)
]	93	5d	135	Llave de cierre inversa
^	94	5e	136	Signo de intercalación (acento circunflejo)
_	95	5f	137	Underscore (raya)
`	96	60	140	Comilla simples de apertura
a	97	61	141	Minúscula a
b	98	62	142	Minúscula b
c	99	63	143	Minúscula c
d	100	64	144	Minúscula d
e	101	65	145	Minúscula e
f	102	66	146	Minúscula f
g	103	67	147	Minúscula g
h	104	68	150	Minúscula h
i	105	69	151	Minúscula i
j	106	6a	152	Minúscula j
k	107	6b	153	Minúscula k

I	108	6c	154	Minúscula I
m	109	6d	155	Minúscula m
n	110	6e	156	Minúscula n
o	111	6f	157	Minúscula o
p	112	70	160	Minúscula p
q	113	71	161	Minúscula q
r	114	72	162	Minúscula r
s	115	73	163	Minúscula s
t	116	74	164	Minúscula t
u	117	75	165	Minúscula u
v	118	76	166	Minúscula v
w	119	77	167	Minúscula w
x	120	78	170	Minúscula x
y	121	79	171	Minúscula y
z	122	7a	172	Minúscula z
{	123	7b	173	Corchete de apertura
	124	7c	174	Línea vertical
}	125	7d	175	Corchete de cierre
~	126	7e	176	Tilde (aproximación)
□	127	7f	177	Borrar (rubout), cross-hatch box

Juego de caracteres extendidos (ANSI)

Carácter	Dec	Hex	Oct	Descripción
€	128	80	200	
	129	81	201	Nota: no es mostrado en este archivo HTML compilado
,	130	82	202	
f	131	83	203	
"	132	84	204	
...	133	85	205	
†	134	86	206	
‡	135	87	207	
^	136	88	210	
%o	137	89	211	
Š	138	8A	212	
‘	139	8B	213	
Œ	140	8C	214	
	141	8D	215	Nota: no es mostrado en este archivo HTML compilado
Ž	142	8E	216	
	143	8F	217	Nota: no es mostrado en este archivo HTML compilado
◆	144	90	220	Nota: no es mostrado en este archivo HTML compilado
‘	145	91	221	
’	146	92	222	
“	147	93	223	
”	148	94	224	

•	149	95	225	
-	150	96	226	
—	151	97	227	
~	152	98	230	
™	153	99	231	
š	154	9A	232	
>	155	9B	233	
œ	156	9C	234	
	157	9D	235	Nota: no es mostrado en este archivo HTML compilado
ž	158	9E	236	
Ÿ	159	9F	237	
	160	A0	240	
í	161	A1	241	
¢	162	A2	242	
£	163	A3	243	
¤	164	A4	244	
¥	165	A5	245	
¦	166	A6	246	
§	167	A7	247	
..	168	A8	250	
©	169	A9	251	
¤	170	AA	252	

«	171	AB	253	
¬	172	AC	254	
	173	AD	255	Nota: no es mostrado en este archivo HTML compilado
®	174	AE	256	
-	175	AF	257	
°	176	B0	260	
±	177	B1	261	
²	178	B2	262	
³	179	B3	263	
'	180	B4	264	
μ	181	B5	265	
¶	182	B6	266	
.	183	B7	267	
,	184	B8	270	
¹	185	B9	271	
º	186	BA	272	
»	187	BB	273	
¼	188	BC	274	
½	189	BD	275	
¾	190	BE	276	
¿	191	BF	277	
À	192	C0	300	

Á	193	C1	301	
Â	194	C2	302	
Ã	195	C3	303	
Ä	196	C4	304	
Å	197	C5	305	
Æ	198	C6	306	
Ҫ	199	C7	307	
Ѐ	200	C8	310	
Ѐ	201	C9	311	
Ӗ	202	CA	312	
Ӗ	203	CB	313	
ӂ	204	CC	314	
Ӵ	205	CD	315	
߱	206	CE	316	
߲	207	CF	317	
߳	208	D0	320	
ߵ	209	D1	321	
߶	210	D2	322	
߷	211	D3	323	
߸	212	D4	324	
߹	213	D5	325	
ߺ	214	D6	326	

x	215	D7	327	
ø	216	D8	330	
Ù	217	D9	331	
Ú	218	DA	332	
Û	219	DB	333	
Ü	220	DC	334	
Ý	221	DD	335	
þ	222	DE	336	
ß	223	DF	337	
à	224	E0	340	
á	225	E1	341	
â	226	E2	342	
ã	227	E3	343	
ä	228	E4	344	
å	229	E5	345	
æ	230	E6	346	
ç	231	E7	347	
è	232	E8	350	
é	233	E9	351	
ê	234	EA	352	
ë	235	EB	353	
ì	236	EC	354	

í	237	ED	355	
â	238	EE	356	
ã	239	EF	357	
ð	240	F0	360	
ñ	241	F1	361	
ò	242	F2	362	
ó	243	F3	363	
ô	244	F4	364	
õ	245	F5	365	
ö	246	F6	366	
÷	247	F7	367	
ø	248	F8	370	
ù	249	F9	371	
ú	250	FA	372	
û	251	FB	373	
ü	252	FC	374	
ý	253	FD	375	
þ	254	FE	376	
ÿ	255	FF	377	

Listado de CLSID

Para su uso con [FileOpenDialog](#), [FileSaveDialog](#), o [FileSelectFolder](#).

Listado Obtenido de [Guía del Registro para Windows en WinGuides Network](#).

Nota: No todos los valores funcionan como parámetro para las funciones de Autolt.

Ejemplo: FileSelectFolder ("Escoger carpeta.", ":{00000000-1080-F9E5-6311-4162E05A6BEE}", 1)

Directorio	Valor para el parámetro de directorio y ficheros
Herramientas Administrativas (Administrative Tools)	":{D20EA4E1-3957-11d2-A40B-0C5020524153}"
Briefcase	":{85BB920-42A0-1069-A2E4-08002B30309D}"
Panel de Control (Control Panel)	":{21EC2020-3AEA-1069-A2DD-08002b30309d}"
Fuentes (Fonts)	":{D20EA4E1-3957-11d2-A40B-0C5020524152}"
Historial (History)	":{FF393560-C2A7-11CF-BFF4-444553540000}"
Inbox	":{00020D75-0000-0000-C000-000000000046}"
Red de Microsoft (Microsoft Network)	":{00028B00-0000-0000-C000-000000000046}"
Mi Computer (My Computer)	":{20D04FE0-3AEA-1069-A2D8-08002B30309D}"
Mis Documentos (My Documents)	":{450D8FBA-AD25-11D0-98A8-0800361B1103}"
Mis Sitios de Red (My Network Places)	":{208D2C60-3AEA-1069-A2D7-08002B30309D}"
Network Computers	":{1f4de370-d627-11d1-ba4f-00a0c91eedba}"
Network Connections	":{7007ACC7-3202-11D1-AAD2-00805FC1270E}"
Impresoras y Faxes (Printers and Faxes)	":{2227A280-3AEA-1069-A2DE-08002B30309D}"
Archivos de Programas (Programs Folder)	":{7be9d83c-a729-4d97-b5a7-1b7313c39e0a}"
Papelera de Reciclaje (Recycle Bin)	":{645FF040-5081-101B-9F08-00AA002F954E}"
Escáner y Cámaras (Scanner and	":{E211B736-43FD-11D1-9EFB-0000F8757FCD}"

Cameras)	
Tareas Programadas (Schedule Tasks)	":::{D6277990-4C6A-11CF-8D87-00AA0060F5BF}"
Carpeta de Menú Inicio (Start Menu Folder)	":::{48e7caab-b918-4e58-a94d-505519c795dc}"
Ficheros Temporales de Internet (Temporary Internet)	":::{7BD29E00-76C1-11CF-9DD0-00A0C9034933}"
Carpeta de Webs (Web Folders)	":::{BDEADF00-C265-11d0-BCED-00A0C90AB50F}"

Estilos de Controles en GUI

Esta página contiene una lista de los estilos corrientes y extendidos que pueden ser usados en ventanas, y también estilos específicos aplicables a controles. Esos estilos se implementan pasando los parámetros "style" y/o "exStyle" en muchas funciones GUI. Para utilizar los valores de referencia a las variables, usted debe agregar la línea #include <GUIConstants.au3> en su Script.

Nota: Estos "valores" son proporcionados simplemente como referencia, es recomendable que siempre utilice el "nombre" del estilo para lograr que sus scripts se mantengan legibles.

Estilos Comunes	Valor	Descripción
		#include <WindowsConstants.au3>
Predefinido/forzado al crear la interfaz gráfica del usuario.		\$GUI_SS_DEFAULT_GUI ver la función GUICreate
\$WS_BORDER	0x00800000	Crea una ventana con un pequeño borde fino
\$WS_POPUP	0x80000000	Crea una ventana popup. Este estilo no puede utilizarse junto con el estilo WS_CHILD
\$WS_CAPTION	0x00C00000	Crea una ventana que tiene una barra de título (incluye el estilo WS_BORDER)
\$WS_CLIPCHILDREN	0x02000000	Excluye el área ocupada por "ventanas hijo" cuando un trazo ocurre dentro de la "ventana padre". Este estilo es aplicado cuando se crea una

		"ventana padre"
\$WS_CLIPSIBLINGS	0x04000000	Sujeta "ventanas hijo" relativas a cualquier otra; es decir, cuando una "ventana hijo" en particular recibe un mensaje WM_PAINT, el estilo WS_CLIPSIBLINGS mantiene todas las otras "ventanas hijo" superpuestas fuera de la región de la "ventana hijo" que va a ser actualizada. Si WS_CLIPSIBLINGS no se especifica y una "ventana hijo" solapa otra, de ser posible, cuando el trazo es dentro del área del cliente, traza en el interior del área del cliente en una "ventana hijo" vecina.
\$WS_DISABLED	0x08000000	Crea una ventana que está inicialmente deshabilitada
\$WS_DLGFRA ME	0x00400000	Crea una ventana que tiene un estilo de borde utilizado típicamente en cuadros de diálogo
\$WS_HSCROLL	0x00100000	Crea una ventana que tiene una barra de desplazamiento horizontal
\$WS_MAXIMIZE	0x01000000	Crea una ventana que se encuentra inicialmente maximizada
\$WS_MAXIMIZEBOX	0x00010000	Crea una ventana que posee un botón para maximizar. No puede combinarse con el estilo WS_EX_CONTEXTHELP. El estilo WS_SYSMENU debe especificarse también
\$WS_MINIMIZE	0x20000000	Crea una ventana que se encuentra inicialmente minimizada
\$WS_MINIMIZEBOX	0x00020000	Crea una ventana que tiene un botón para minimizar. No puede combinarse con el estilo WS_EX_CONTEXTHELP. El estilo WS_SYSMENU debe especificarse también
\$WS_OVERLAPPED	0x00000000	Crea una ventana superpuesta. Tiene una barra de título y un borde. Lo mismo que el estilo WS_TILED.
\$WS_OVERLAPPEDWINDOW	0x00CF0000	Crea una ventana superpuesta con los estilos WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU,

		WS_THICKFRAME, WS_MINIMIZEBOX, y WS_MAXIMIZEBOX. Igual que el estilo WS_TILEDWINDOW
\$WS_POPUPWINDOW	0x80880000	Crea una ventana popup con los estilos WS_BORDER, WS_POPUP, y WS_SYSMENU. Los estilos WS_CAPTION y WS_POPUPWINDOW deben ser combinados para hacer que el menú de la ventana esté visible
\$WS_SIZEBOX	0x00040000	Crea una ventana que tiene un borde para dimensionarla. Lo mismo que con el estilo WS_THICKFRAME
\$WS_SYSMENU	0x00080000	Crea una ventana que tiene un menú de ventana sobre la barra de título. El estilo WS_CAPTION debe especificarse también
\$WS_THICKFRAME	0x00040000	Crea una ventana que tiene un borde para dimensionarla. Lo mismo que con el estilo WS_SIZEBOX
\$WS_VSCROLL	0x00200000	Crea una ventana que tiene una barra de desplazamiento vertical
\$WS_VISIBLE	0x10000000	Crea una ventana que inicialmente es visible
\$WS_CHILD	0x40000000	Crea una "ventana hijo". Una ventana de este estilo no puede tener una barra de menú. No puede ser combinada con el estilo WS_POPUP
\$WS_GROUP	0x00020000	Especifica cual es el primer control de un grupo de controles. El grupo consistirá del primer control, adelantando a próximos controles definidos después de él, con el estilo WS_GROUP
\$WS_TABSTOP	0x00010000	Convierte al control en una detención Tab, lo cual posibilita al usuario seleccionarlo mediante tabulación entre otros controles de un cuadro de diálogo
\$DS_MODALFRAME	0x00000080	Crea un cuadro de diálogo modal en un marco, el cual puede combinarse con una barra de título y un menú de ventana, especificando los estilos

		WS_CAPTION y WS_SYSMENU
\$DS_SETFOREGROUND	0x000000200	Este estilo es útil para cuadros de diálogo modal que requieren una atención inmediata de parte del usuario indiferente, si la propia ventana de la aplicación está fuera del primer plano
\$DS_CONTEXTHELP	0x00002000	Añade un signo de pregunta en la barra de título del cuadro de diálogo. No puede ser usado con los estilos WS_MAXIMIZEBOX o WS_MINIMIZEBOX. Lo mismo con el estilo extendido WS_EX_CONTEXTHELP

Estilos Extendidos Comunes	Valor	Descripción
		#include <WindowsConstants.au3>
\$WS_EX_ACCEPTFILES	0x00000010	Habilita un control editor o de ingreso en la ventana GUI creada, a recibir nombres de archivo vía arrastrar y soltar. El control debe tener también el estado \$GUI_DROPACCEPTED seleccionado mediante GUICtrlSetState
\$WS_EX_APPWINDOW	0x00040000	Fuerza una ventana visible al nivel superior sobre la barra de tareas
\$WS_EX_COMPOSITED	0x02000000	Windows XP o superior: dibuja todos los descendientes de una ventana desde la parte inferior a la superior usando un doble buffer.
\$WS_EX_CLIENTEDGE	0x00000200	Especifica que los límites para el borde de la ventana tendrán el aspecto de hundidos
\$WS_EX_CONTEXTHELP	0x00000400	Agrega un signo de interrogación en la barra de título de la ventana. No puede ser combinado con WS_MAXIMIZEBOX o WS_MINIMIZEBOX
\$WS_EX_DLGMODALFRAME	0x00000001	Crea una ventana que tiene un borde doble; la ventana puede, opcionalmente, crearse con

		una barra de título al especificar el estilo WS_CAPTION en el parámetro dwStyle
\$WS_EX_MDICHILD	0x00000040	Crea una "ventana hijo" que se moverá con la ventana padre. (simulación de una ventana MDI con maximizar/minimizar no son simuladas)
\$WS_EX_OVERLAPPEDWINDOW	0x00000300	Combina los estilos WS_EX_CLIENTEDGE y WS_EX_WINDOWEDGE
\$WS_EX_STATICEDGE	0x00020000	Crea una ventana con un estilo de borde tridimensional, proyectado para utilizarse con ítems que no aceptan ingreso desde el usuario
\$WS_EX_TOPMOST	0x00000008	Especifica que una ventana creada con este estilo deberá estar colocada encima de todas las demás que no tengan esta característica y mantenerse en la misma condición, aun cuando la ventana esté inactiva
\$WS_EX_TRANSPARENT	0x00000020	La ventana aparece transparente porque los segmentos en las ventanas hermanas subyacentes ya han sido rellenados
\$WS_EX_TOOLWINDOW	0x00000080	Crea una ventana de herramienta; lo que significa que es una ventana pensada para utilizarse como barra de herramienta flotante. Una ventana de herramienta tiene una barra de título que es más estrecha que una barra de título normal, y el título de la ventana figura con una fuente algo más pequeña. Una ventana de herramienta no aparecerá en la barra de tareas o en el cuadro de diálogo que se muestra cuando el usuario presiona la combinación ALT+TAB. Si una ventana de herramientas incluye un menú de sistema, el ícono no será emplazado en la barra de título. Sin embargo, puede mostrarlo utilizando el método ALT+SPACE
\$WS_EX_WINDOWEDGE	0x00000100	Establece que la ventana tiene un borde cuyo límite aparece elevado

\$WS_EX_LAYERED	0x00080000	Crea una ventana de capas. Tenga en cuenta que no puede ser utilizada con ventanas hijo
\$GUI_WS_EX_PARENTDRAG	0x00100000	Permite a un control de etiqueta o imagen ser empleado como la barra de título para arrastrar al conjunto de la ventana padre

Estilos de Caja de Chequeo (Checkbox)	Valor	Descripción
		#include <ButtonConstants.au3>
Predefinido/forzado		Ver función GUICtrlCreateCheckbox
\$BS_3STATE	0x0005	Crea una caja de chequeo (check box) que puede ser inhabilitada así como seleccionada o limpiada. Use el estado 'no disponible' para mostrar cual estado de la caja de chequeo no está determinado
\$BS_AUTO3STATE	0x0006	Crea una caja de chequeo de tres estados, la cual puede alternar entre los estados: marcado, chequeado y deseleccionado cada vez que el usuario clickea sobre el control
\$BS_AUTOCHECKBOX	0x0003	Crea una caja de chequeo en la que el estado de chequeo alterna entre seleccionado y deseleccionado cada vez que el usuario clickea sobre el control
\$BS_CHECKBOX	0x0002	Crea una caja de chequeo pequeña y 'limpia' con la etiqueta alineada a la derecha. Para alinear el texto a la izquierda, combine esta bandera (flag) con el estilo BS_RIGHTBUTTON
\$BS_LEFT	0x0100	Alineación a la izquierda del texto contenido dentro de una caja de chequeo
\$BS_PUSHLIKE	0x1000	Crea un botón (como una caja de chequeo, caja de chequeo de tres estados, o un botón circular (radio button) semejante a un botón de pulsado. El botón es oprimido cuando la caja de chequeo está marcada y

		liberado cuando es desmarcada
\$BS_RIGHT	0x0200	Alineación a la derecha del texto contenido dentro de una caja de chequeo
\$BS_RIGHTBUTTON	0x0020	Posiciona una caja de chequeo a la derecha del rectángulo de un botón
\$BS_GROUPBOX	0x0007	Crea un rectángulo en el cual otros botones puedan estar agrupados. Cualquier texto asociado con este estilo es mostrado en la esquina superior izquierda del rectángulo
\$BS_AUTORADIOBUTTON	0x0009	Igual que un botón circular, excepto cuando el usuario lo selecciona, el botón se destaca automáticamente y quita la selección de cualquier otro botón circular con el mismo estilo dentro del mismo grupo

Estilos de Botón	Valor	Descripción
		#include <ButtonConstants.au3>
Predefinido/forzado		Ver funciones GUICtrlCreateButton , GUICtrlCreateCheckbox , GUICtrlCreateRadio
\$BS_BOTTOM	0x0800	Coloca el texto en el fondo del rectángulo del botón.
\$BS_CENTER	0x0300	Centra horizontalmente el texto dentro del rectángulo del botón.
\$BS_DEFPUSHBUTTON	0x0001	Muestra un borde oscuro alrededor del rectángulo del botón. Si el botón está en una caja de diálogo (dialog box), el usuario puede seleccionar el botón oprimiendo la tecla ENTER, aunque el botón no tenga el foco. Este estilo es muy útil para permitir al usuario seleccionar rápidamente la opción más probable, o por defecto
\$BS_MULTILINE	0x2000	Habilita la escritura en múltiples líneas del texto dentro del rectángulo del botón en caso de que el texto sea demasiado largo

\$BS_TOP	0x0400	Coloca el texto en la parte superior del rectángulo del botón
\$BS_VCENTER	0x0C00	Centra verticalmente el texto dentro del rectángulo del botón
\$BS_ICON	0x0040	Especifica que el botón tendrá un ícono
\$BS_BITMAP	0x0080	Especifica que el botón mostrará una imagen.
\$BS_FLAT	0x8000	Especifica que el botón es bidimensional; no utiliza el efecto de sombreado para crear una botón tridimensional
\$BS_NOTIFY	0x4000	Capacita a un botón para enviar mensajes de notificación BN_KILLFOCUS y BN_SETFOCUS a la ventana padre. Note que los botones envían el mensaje de notificación BN_CLICKED tengan o no este estilo. Para obtener el mensaje de notificación BN_DBCLK, el botón debe tener un estilo BS_RADIOBUTTON o BS_OWNERDRAW.

Estilos Cajas de selección (Combo)	Valor	Descripción
		#include <ComboConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_COMBO Ver función GUICtrlCreateCombo .
\$CBS_AUTOHSCROLL	0x0040	Automáticamente desplaza el texto a la derecha cuando el usuario teclea un carácter al final de la línea. Si este estilo no es establecido, solo se permite el texto dentro del límite del rectángulo del control
\$CBS_DISABLENOSCROLL	0x0800	Muestra una barra de desplazamiento (scroll bar) deshabilitada en la caja de texto cuando la caja no contiene ningún ítem para desplazar. Sin este estilo, la barra de desplazamiento es ocultada cuando la caja de texto no contiene ningún artículo
\$CBS_DROPDOWN	0x0002	Muestra la caja de texto por defecto. El usuario puede ver el contenido de la caja de texto a través de un

		pequeño ícono en la esquina derecha del control
\$CBS_DROPDOWNLIST	0x0003	Muestra un campo de texto estático que muestra el ítem actual seleccionado en el control
\$CBS_LOWERCASE	0x4000	Convierte a minúscula cualquier texto en mayúscula dentro del control
\$CBS_NOINTEGRALHEIGHT	0x0400	Especifica que un combo box tendrá exactamente las mismas dimensiones establecidas por la aplicación cuando es creado. Usualmente, Windows CE dimensiona un combo box para no mostrar ítem parciales
\$CBS_OEMCONVERT	0x0080	Convierte el texto tecleado en el edit de un combo box del juego de caracteres de Windows CE para el juego de caracteres OEM solamente. Este estilo es muy útil cuando el combo box contiene nombre de ficheros. Esto es aplicable solo a combo boxes creados con el estilo CBS_DROPDOWN.
\$CBS_SIMPLE	0x0001	Muestra la caja de texto en todo momento. La actual selección del combo box es mostrada en esta caja de texto
\$CBS_SORT	0x0100	Ordena alfabéticamente las cadenas de texto introducidas en el combo box
\$CBS_UPPERCASE	0x2000	Convierte a mayúscula cualquier texto en minúscula dentro del control.

Estilos de Lista (List)	Valor	Descripción
		#include <ListboxConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_LIST ver función GUICtrlCreateList .
\$LBS_DISABLENOSCROLL	0x1000	Muestra una barra de desplazamiento (scroll bar) deshabilitada en la caja de lista cuando la caja no

		contiene ningún ítem para desplazar. Sin este estilo, la barra de desplazamiento es ocultada cuando la caja de lista no contiene ningún ítem
\$LBS_NOINTEGRALHEIGHT	0x0100	Especifica que una caja de lista tendrá exactamente las misma dimensiones establecidas por la aplicación cuando es creada
\$LBS_NOSEL	0x4000	Especifica que el usuario puede ver los ítem listados pero no seleccionarlos
\$LBS_NOTIFY	0x0001	Notifica a la ventana padre cuando se hace click o doble click sobre una cadena de texto en la caja de lista
\$LBS_SORT	0x0002	Ordena alfabéticamente las cadena de texto en la caja de lista
\$LBS_STANDARD	0xA00003	Ordena las cadenas de texto alfabéticamente en la caja de lista. La ventana padre recibe un mensaje de entrada (input message)cuando el usuario clickea o hace doble click en una cadena de texto. La caja de lista es bordeada totalmente. (LBS_NOTIFY LBS_SORT WS_VSCROLL WS_BORDER)
\$LBS_USETABSTOPS	0x0080	Permite a la caja de lista reconocer y expandir caracteres tab cuando es entrada una cadena de texto. Por defecto las posiciones tab son de 32 unidades de caja. Una unidad de caja de diálogo es igual a un cuarto de la unidad base del ancho de la caja de diálogo actual

Estilos Editor/Ingreso	Valor	Descripción
		#include <EditConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_EDIT, \$GUI_SS_DEFAULT_INPUT ver GUICtrlCreateEdit o la función GUICtrlCreateInput

\$ES_AUTOHSCROLL	0x0080	Automáticamente desplaza el texto a la derecha por 10 caracteres cuando el usuario clica un carácter al final de la línea. Cuando el usuario presiona la tecla ENTER, el control de desplazamiento ubica el texto en la línea inmediata inferior de la actual
\$ES_AUTOVSCROLL	0x0040	Desplaza el texto para una nueva línea cuando es oprimida la tecla ENTER o llega al final de la línea
\$ES_CENTER	0x0001	Centra el texto en un control con multilínea
\$ES_LOWERCASE	0x0010	Convierte todos los caracteres a minúscula cuando estos son tecleados al control.
\$ES_NOHIDESEL	0x0100	Niega el comportamiento por defecto de un control de edición. Por defecto el control de edición oculta el texto seleccionado cuando el control pierde el foco e invierte la selección cuando el control de edición recupera el foco de entrada. Si usted utiliza ES_NOHIDESEL, la selección del texto es invertida, aunque el control no tenga el foco
\$ES_NUMBER	0x2000	Acepta solo dígitos numéricos para ser tecleados dentro de un control
\$ES_OEMCONVERT	0x0400	Estilo muy útil cuando se muestra nombres de ficheros en el control
\$ES_MULTILINE	0x0004	Designa un control multilínea. Por defecto el control de texto no incorpora este estilo
\$ES_PASSWORD	0x0020	Muestra asteriscos (*) por cada carácter tecleado dentro del control
\$ES_READONLY	0x0800	Previene que el usuario pueda editar el contenido de un control
\$ES_RIGHT	0x0002	Alineación a la derecha del texto de un control
\$ES_UPPERCASE	0x0008	Convierte todos los caracteres a minúscula cuando estos son tecleados al control.
\$ES_WANTRETURN	0x1000	Especifica que puede usarse el retorno de línea cuando el usuario oprime la tecla ENTER mientras teclea texto en un control multilínea dentro de una caja de diálogo. Si no se especifica este estilo, presionando la tecla ENTER no provocará

		un retorno de línea, será semejante a presionar un botón. Este estilo no tiene efecto en controles que no son multilínea
--	--	--

Estilos Barra de Progreso	Valor	Descripción
		#include <ProgressConstants.au3>
\$PBS_SMOOTH	0x01	Muestra el estado del progreso en una barra de desplazamiento lisa, en lugar de la barra segmentada que es la predefinida
\$PBS_VERTICAL	0x04	Muestra el estado del progreso verticalmente, en sentido inferior al superior

Estilos Up-down	Valor	Descripción
		#include <UpdownConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_UPDOWN ver la función GUICtrlCreateUpdown .
\$UDS_ALIGNLEFT	0x08	Posiciona el control up-down consecutivo al margen izquierdo de la ventana acompañada. Esta se desplaza a la derecha y el ancho es reducido en orden de reacomodarse al tamaño del control up-down
\$UDS_ALIGNRIGHT	0x04	Posiciona el control up-down consecutivo al margen derecho de la ventana acompañada. El ancho es reducido en orden de reacomodarse al tamaño del control up-down
\$UDS_ARROWKEYS	0x20	El control up-down procesa los comandos desde las teclas direccionales FLECHA ARRIBA y FLECHA ABAJO del teclado
\$UDS_HORZ	0x40	Las flechas direccionales del control up-down estarán colocadas en el eje izquierda/derecha en lugar de arriba y abajo

\$UDS_NOTHOUSANDS	0x80	Impide la inserción de puntos separadores por cada tres posiciones decimales
\$UDS_WRAP	0x01	Establece que si el valor contenido es incrementado o restado más allá de los valores máximo y mínimo, retomará al valor inicial/final según sea el caso

Estilos de Etiqueta/Estático	Valor	Descripción
		#include <StaticConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_LABEL, \$GUI_SS_DEFAULT_ICON, \$GUI_SS_DEFAULT_PIC. Ver funciones GUICtrlCreateLabel , GUICtrlCreateIcon , GUICtrlCreatePic
\$SS_BLACKFRAME	0x07	Especifica una casilla con un cuadro trazado en el mismo color que el marco de la ventana. Este color es negro en el esquema predefinido
\$SS_BLACKRECT	0x04	Especifica un rectángulo lleno con el color actual del marco de la ventana. Este color es negro en el esquema predefinido
\$SS_CENTER	0x01	Especifica un rectángulo simple y centra el valor del texto erróneo en el rectángulo. El control automáticamente recorta las palabras que se extienden pasando el final de cada línea y las coloca al comienzo de la próxima línea centrada
\$SS_CENTERIMAGE	0x0200	Especifica que el punto medio de un control estático con el estilo SS_BITMAP permanecerá inmóvil aun cuando el control sea redimensionado. Los cuatro lados son ajustados a reposicionar el nuevo mapa de bits. Si el mapa de bits es más pequeño que el área de cliente del control, el resto del área del cliente es ocupada con el color del pixel en la esquina superior izquierda del mapa de bits. Puede utilizarse con un control estático que contenga solamente una línea de texto. Microsoft Windows XP: Este estilo ya no contempla el resultante de las porciones desocupadas del control con

		el relleno de color del píxel de la esquina superior izquierda del mapa de bits o ícono. Las porciones sin ocupar en el control permanecerán entonces con el color de relleno del fondo
\$SSETCHEDFRAME	0x12	Traza el marco del control estático usando el estilo de borde EDGEETCHED
\$SSETCHEDHORZ	0x10	Traza el límite superior e inferior del control estático utilizando el estilo de borde EDGEETCHED
\$SSETCHEDVERT	0x11	Traza el límite izquierdo y derecho del control estático utilizando el estilo de borde EDGEETCHED
\$SSGRAYFRAME	0x08	Establece un casillero con el marco trazado en el mismo color que el relleno de fondo de la pantalla (escritorio). Este color es gris en el esquema de color predefinido
\$SSGRAYRECT	0x05	Especifica un rectángulo lleno con el color del fondo de la pantalla actual. Este color es gris en el esquema de color predefinido
\$SSLEFT	0x0000	Especifica un rectángulo simple y alinea a la izquierda el texto en el rectángulo. El texto es formateado antes de ser mostrado. Las palabras que excedan el largo de la línea son colocadas automáticamente al comienzo de la siguiente. Si son más las líneas que las que entran en el alto del control, las palabras que excedan del largo de la última línea visible serán truncadas
\$SSLEFTNOWORDWRAP	0x0C	Especifica un rectángulo y alinea el texto interno a la izquierda. Las tabulaciones son expandidas pero las palabras no son pasadas a la siguiente línea. Aquellas que excedan el largo de cada línea serán truncadas
\$SSNOPREFIX	0x80	Previene la interpretación de cualquier carácter (&) en el texto del control como un prefijo de acceso abreviado al mismo. Una aplicación puede combinar el estilo SS_NOPREFIX con otros, utilizando el operador de dígito binario selectivo OR (). Esto puede resultar útil cuando nombres de archivos u otras cadenas que puedan contener un signo (&) deban ser mostrados con un control de etiqueta

		en un cuadro de dialogo
\$SS_NOTIFY	0x0100	Envía a la ventana padre la notificación STN_CLICKED cuando el usuario hace click sobre el control
\$SS_RIGHT	0x0002	Establece el texto a la derecha en el interior del rectángulo especificado
\$SS_RIGHTJUST	0x0400	La esquina inferior derecha del control estático con el estilo SS_BITMAP o SS_ICON permanecerá fija cuando el control es redimensionado. Solamente los lados superior e izquierdo serán ajustados a fin de acomodar el nuevo mapa de bits o ícono
\$SS_SIMPLE	0x0B	Especifica un rectángulo simple y muestra una línea única de texto que permanece alineada a la izquierda. La línea de texto no podrá ser ordenada o alterada de manera alguna. Además, si el control es desactivado, el texto en su interior no será colocado en el color gris típico.
\$SS_SUNKEN	0x1000	Traza un borde con aspecto de semi-profundidad alrededor del control estático.
\$SS_WHITEFRAME	0x09	Establece una casilla con el marco trazado en el color de fondo de la ventana. Este color es blanco en el esquema predefinido.
\$SS_WHITERECT	0x06	El rectángulo es ocupado con el color de fondo actual de la ventana. Este color es blanco en el esquema predefinido.

Estilo de Pestaña (Tab)	Valor	Descripción
		#include <TabConstants.au3>
Predefinido/forzado		Ver función GUICtrlCreateTab .
\$TCS_SCROLLTOPPOSITE	0x0001	Barras de desplazamiento innecesarias en el lado opuesto a la pestaña seleccionada

\$TCS_BOTTOM	0x0002	Las pestañas aparecen al fondo del control. Este valor es igual a TCS_RIGHT. Este estilo no es soportado si usted usa ComCtl32.dll versión 6
\$TCS_RIGHT	0x0002	Las pestañas aparecen verticalmente al lado derecho del control que usen el estilo TCS_VERTICAL. Este valor es igual a TCS_BOTTOM. Este estilo no es soportado si se emplean estilos visuales
\$TCS_MULTISELECT	0x0004	Múltiples pestañas pueden ser seleccionadas usando la tecla CTRL. Este estilo debe ser usado con el estilo TCS_BUTTONS
\$TCS_FLATBUTTONS	0x0008	Las pestañas seleccionadas aparecen como marcadas hacia dentro con relación a aquellas que no lo están. Este estilo solo afecta a los controles de pestaña con el estilo TCS_BUTTONS
\$TCS_FORCEICONLEFT	0x0010	Los iconos aparecen alineados al borde izquierdo de cada pestaña con ancho fijo. Este estilo solo puede ser usado con el estilo TCS_FIXEDWIDTH
\$TCS_FORCELABELLEFT	0x0020	Las etiquetas son alineadas al borde izquierdo de cada pestaña con ancho fijo; esto es, la etiqueta es exhibida inmediatamente a la derecha del ícono en lugar de estar centrada. Este estilo solo puede usarse con el estilo TCS_FIXEDWIDTH, y este implica el estilo TCS_FORCEICONLEFT
\$TCS_HOTTRACK	0x0040	Los ítems bajo el cursor son automáticamente resaltados
\$TCS_VERTICAL	0x0080	Las pestañas aparecen por el lado izquierdo del control, con el texto de la pestaña exhibido verticalmente. Este estilo es válido sólo cuando es usado con el estilo TCS_MULTILINE. Para hacer que las pestañas aparezcan al lado derecho del control, también use el estilo TCS_RIGHT. Este estilo no es soportado si usted usa la versión ComCtl32.dll versión 6

\$TCS_TABS	0x0000	Las pestañas aparecen como pestañas, y un borde se dibuja alrededor del área de despliegue. Este estilo es por defecto
\$TCS_BUTTONS	0x0100	Las pestañas aparecen como botones, sin un borde dibujado alrededor del área de despliegue
\$TCS_SINGLELINE	0x0000	Solo una fila de pestañas es mostrada. El usuario debe desplazarse para ver más pestañas, si es necesario. Este estilo es por defecto
\$TCS_MULTILINE	0x0200	Múltiples filas de pestañas es mostrada , de ser necesario, de esta forma todas las pestañas con visibles
\$TCS_RIGHTJUSTIFY	0x0000	El ancho de cada pestaña es incrementada, si es necesario, a fin de que cada fila de pestañas llena la anchura entera del control pestaña. Este estilo de Windows es ignorado a menos que el estilo TCS_MULTILINE sea especificado
\$TCS_FIXEDWIDTH	0x0400	Todas las pestañas tienen el mismo ancho. Este estilo no puede ser combinado con el estilo TCS_RIGHTJUSTIFY
\$TCS_RAGGEDRIGHT	0x0800	Las filas de pestañas no se alargan para cubrir todo el control. Este estilo es por defecto
\$TCS_FOCUSONBUTTONDOWN	0x1000	El control de pestaña recibe el foco cuando es clickeado
\$TCS_OWNERDRAWFIXED	0x2000	La ventana padre es responsable de redibujar las pestañas
\$TCS_TOOLTIPS	0x4000	El control de pestaña coloca un control tooltip asociado a la misma
\$TCS_FOCUSNEVER	0x8000	El control de pestaña no recibe el foco cuando es clickeada

Estilos de Clip Avi	Valor	Descripción
		#include <AVIConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_AVI ver función GUICtrlCreateAvi .
\$ACS_AUTOPLAY	0x04	Comienza la reproducción de la animación tan pronto como el clip AVI es abierto
\$ACS_CENTER	0x01	Centra la animación en la ventana del control de animación
\$ACS_TRANSPARENT	0x02	Le permite capturar el color de fondo de la ventana subyacente a la animación, simulando un efecto de transparencia. (Valor predefinido)
\$ACS_NONTRANSPARENT	0x10	Substituye el estilo predefinido ACS_TRANSPARENT

Estilos Fecha	Valor	Descripción
		#include <DateTimeConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_DATE ver función GUICtrlCreateDate
\$DTS_UPDOWN	0x01	Ubica un control UpDown a la derecha del control DTP para modificar los valores de tiempo. Este estilo puede ser usado en lugar de la lista desplegable en calendario mensual, como estilo por defecto
\$DTS_SHOWNONE	0x02	Habilita el control para aceptar valores de "no fecha" como un estado de selección válida
\$DTS_LONGDATEFORMAT	0x04	Muestra la fecha en formato extendido. El formato por defecto para este estilo es definido por LOCALE_SLONGDATEFORMAT, que produce como resultado: "Viernes, abril 19, 1998."
\$DTS_TIMEFORMAT	0x09	Muestra la hora. El formato por defecto para este estilo es definido por LOCALE_STIMEFORMAT, que produce como resultado:

		"5:31:42 PM."
\$DTS_RIGHTALIGN	0x20	El calendario mensual es alineado a la derecha con rl control en lugar de alinearse a la izquierda por defecto
\$DTS_SHORTDATEFORMAT	0x00	Muestra la fecha en formato corto. El formato por defecto para este estilo es definido por LOCALE_SSHORTDATE, que produce como resultado: "4/19/96"

Estilos MesCalendario	Valor	Descripción
		#include <DateTimeConstants.au3>
Predefinido/forzado		Ver función GUICtrlCreateMonthCal .
\$MCS_NOTODAY	0x10	El control mes-calendario mostrará la fecha "hoy" al final del control
\$MCS_NOTODAYCIRCLE	0x08	El control mes-calendario no tendrá un círculo rodeando la fecha de "hoy"
\$MCS_WEEKNUMBERS	0x04	El control mes-calendario mostrará el número de semanas (1-52) a la izquierda de cada columna de días. La semana 1 es establecida como la primera semana que contiene al menos cuatro días

Estilos control de árbol TreeView	Valor	Descripción
		#include <TreeviewConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_TREEVIEW ver GUICtrlCreateTreeView
\$TVS_HASBUTTONS	0x0001	Muestra un botón 'más' (+) y 'menos' (-) en la cabecera de ítems padres. Un click sobre estos botones expande o contrae la lista de ítems hijos. Para enlazar la lista al

		inicio principal del árbol usar: TVS_LINESATROOT , el cual también debe ser especificado
\$TVS_HASLINES	0x0002	Uso de líneas para señalar el nivel jerárquico de la lista de ítems
\$TVS_LINESATROOT	0x0004	Uso de líneas para enlazar las cabeceras de la lista. Este valor es ignorado si TVS_HASLINES no es especificado
\$TVS_DISABLEDRAGDROP	0x0010	Previene el control de árbol para recibir notificación de mensajes TVN_BEGINDRAG
\$TVS_SHOWSELALWAYS	0x0020	Posibilita que un ítems no deje de estar seleccionado aun cuando el control pierda el foco
\$TVSRTLREADING	0x0040	Las ventanas normales muestran el texto de izquierda a derecha (LTR). Windows puede usar una máscara para mostrar lenguajes como el Hebreo o el Arábico que leen de derecha a izquierda (RTL). Normalmente , el control de árbol mostrará el texto de acuerdo con el formato de la ventana. Si TVSRTLREADING es establecido, el control de árbol lee en la dirección opuesta propuesta por el formato de la ventana
\$TVSNOTOOTIPS	0x0080	El control de árbol no soporta mensajes flotantes (tooltips)
\$TVSCHECKBOXES	0x0100	Habilita cajas de chequeo para los ítems en el control tree-view control. Uno control tree-view es creado con este estilo, este estilo no puede ser removido. en su lugar, usted puede destruir el control y crearlo de nuevo en el mismo lugar.
\$TVSTRACKSELECT	0x0200	Habilita el subrayado de los ítems seleccionados
\$TVSSINGLEEXPAND	0x0400	Cuando este estilo es habilitado, los cambios de selección en el control de árbol posibilitan que los ítems seleccionados se expandan mientras que los ítems deseleccionados se contraigan simultáneamente. Si el mouse es usado para Simple-click el ítems es seleccionado y expandido. Si el usuario oprime la tecla CTRL mientras el ítem mientras selecciona un ítems, este se desmarca pero no se contrae la rama

\$TVS_FULLROWSELECT	0x1000	Habilita la selección en todas las filas del control de árbol. La fila completa de los ítems seleccionados es marcada, u cualquier click sobre un ítem de la fila causa que este sea seleccionado. Este estilo no puede ser usado en conjunción con el estilo TVS_HASLINES
\$TVS_NOSCROLL	0x2000	Deshabilita el desplazamiento horizontal en el control. El control no muestra ninguna barra de desplazamiento horizontal
\$TVS_NONEVENHEIGHT	0x4000	La altura de los ítems puede ser establecida con el mensaje TVM_SETITEMHEIGHT . Por defecto, la altura de los ítems es común

Estilos Slider	Valor	Descripción
		#include <SliderConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_SLIDER Ver función GUICtrlCreateSlider .
\$TBS_AUTOTICKS	0x0001	Agrega marcas de señalización cuando se establece el rango sobre la barra de desplazamiento utilizando el mensaje TBM_SETRANGE
\$TBS_BOTH	0x0008	Coloca marcas en ambos lados de la barra de desplazamiento
\$TBS_BOTTOM	0x0000	Coloca marcas en la parte inferior de una barra de desplazamiento horizontal
\$TBS_HORZ	0x0000	Establece la barra de desplazamiento en posición horizontal. Este es el estilo predefinido
\$TBS_VERT	0x0002	Establece que la barra de desplazamiento se ubicará en posición vertical
\$TBS_NOTHUMB	0x0080	Especifica que la barra de desplazamiento no tiene control para deslizamiento
\$TBS_NOTICKS	0x0010	Especifica que no se colocarán marcas en la barra de desplazamiento

\$TBS_LEFT	0x0004	Coloca marcas en el lado izquierdo de una barra de desplazamiento vertical
\$TBS_RIGHT	0x0000	Coloca marcas en el lado derecho de una barra de desplazamiento vertical
\$TBS_TOP	0x0004	Coloca marcas en la parte superior de una barra de desplazamiento horizontal

Estilos ListView	Valor	Descripción
		#include <ListviewConstants.au3>
Predefinido/forzado		\$GUI_SS_DEFAULT_LISTVIEW ver función GUICtrlCreateListView .
\$LVS_ICON	0x0000	Este estilo especifica vista en iconos
\$LVS_REPORT	0x0001	Este estilo especifica vista en reporte
\$LVS_SMALLICON	0x0002	Este estilo especifica vista en iconos pequeños
\$LVS_LIST	0x0003	Este estilo especifica vista en lista
\$LVS_EDITLABELS	0x0200	Los ítem de texto pueden ser editados manualmente
\$LVS_NOCOLUMNHEADER	0x4000	Las cabeceras de columnas no muestran una vista de reporte. Por defecto, las cabeceras de las columnas (títulos) muestran una vista de reporte
\$LVS_NOSORTHEADER	0x8000	Las cabeceras de columnas no son tratadas como botones. Este estilo puede ser usado si se quiere que al clickear una cabecera de columna no muestra ninguna acción, como ordenar la lista
\$LVS_SINGLESEL	0x0004	Solo un ítem puede ser seleccionado a la vez
\$LVS_SHOWSELALWAYS	0x0008	La selección, si la hay, se muestra siempre, aunque el control no reciba el foco

\$LVS_SORTASCENDING	0x0010	El índice de ítems es ordenado basado en el texto del ítem de modo ascendente
\$LVS_SORTDESCENDING	0x0020	El índice de ítems es ordenado basado en el texto del ítem de modo descendente

Estilos Extendidos ListView	Valor	Descripción
		#include <ListviewConstants.au3>
\$LVS_EX_FULLROWSELECT	0x00000020	Cuando un ítem es seleccionado, el ítem y todos sus subítems serán marcados también
\$LVS_EX_GRIDLINES	0x00000001	Muestra líneas de tabulación alrededor de los ítems y subítems.
\$LVS_EX_HEADERDRAGDROP	0x00000010	Habilita ordenamiento de modo "arrastra y suelta" las columnas en un control de lista
\$LVS_EX_TRACKSELECT	0x00000008	Habilita la selección de ítem-hipervínculos en un control de lista. La selección de Item-hipervínculos significa que el ítems se marcará automáticamente después de haber colocado sobre el mismo el cursor por un tiempo determinado
\$LVS_EX_CHECKBOXES	0x00000004	Habilita cajas de chequeos en los ítems de un control de lista.
\$LVS_EX_BORDERSELECT	0x00008000	Si este estilo es usado, cuando el ítem es seleccionado los colores del borde cambian para delimitarlo
\$LVS_EX_DOUBLEBUFFER	0x00010000	
\$LVS_EX_FLATSB	0x00000100	Habilita una barra de desplazamiento plana en el control de lista
\$LVS_EX_MULTIWORKAREAS	0x00002000	

\$LVS_EX_SNAPTOGRID	0x00080000	
\$LVS_EX_SUBITEMIMAGES	0x00000002	Permite que se muestren imagenes en lugar de subitems

@OSLang codes

Possible return values (strings) of [@OSLang](#)

List was generated from [Windows 2000 - List of Locale IDs and Language Groups](#)

Note: Codes that contain letters could possibly have the letters in uppercase.

Code	Meaning
0436	Afrikaans
041c	Albanian
0401	Arabic_Saudi_Arabia
0801	Arabic_Iraq
0c01	Arabic_Egypt
1001	Arabic.Libya
1401	Arabic_Algeria
1801	Arabic_Morocco
1c01	Arabic_Tunisia
2001	Arabic_Oman
2401	Arabic_Yemen
2801	Arabic_Syria
2c01	Arabic_Jordan
3001	Arabic_Lebanon

3401	Arabic_Kuwait
3801	Arabic_UAE
3c01	Arabic_Bahrain
4001	Arabic_Qatar
042b	Armenian
042c	Azeri_Latin
082c	Azeri_Cyrillic
042d	Basque
0423	Belarusian
0402	Bulgarian
0403	Catalan
0404	Chinese_Taiwan
0804	Chinese_PRC
0c04	Chinese_Hong_Kong
1004	Chinese_Singapore
1404	Chinese_Macau
041a	Croatian
0405	Czech
0406	Danish
0413	Dutch_Standard
0813	Dutch_Belgian
0409	English_United_States

0809	English_United_Kingdom
0c09	English_Australian
1009	English_Canadian
1409	English_New_Zealand
1809	English_Irish
1c09	English_South_Africa
2009	English_Jamaica
2409	English_Caribbean
2809	English_Belize
2c09	English_Trinidad
3009	English_Zimbabwe
3409	English_Philippines
0425	Estonian
0438	Faeroese
0429	Farsi
040b	Finnish
040c	French_Standard
080c	French_Belgian
0c0c	French_Canadian
100c	French_Swiss
140c	French_Luxembourg
180c	French_Monaco

0437	Georgian
0407	German_Standard
0807	German_Swiss
0c07	German_Austrian
1007	German_Luxembourg
1407	German_Liechtenstei
408	Greek
040d	Hebrew
0439	Hindi
040e	Hungarian
040f	Icelandic
0421	Indonesian
0410	Italian_Standard
0810	Italian_Swiss
0411	Japanese
043f	Kazakh
0457	Konkani
0412	Korean
0426	Latvian
0427	Lithuanian
042f	Macedonian
043e	Malay_Malaysia

083e	Malay_Brunei_Darussalam
044e	Marathi
0414	Norwegian_Bokmal
0814	Norwegian_Nynorsk
0415	Polish
0416	Portuguese_Brazilian
0816	Portuguese_Standard
0418	Romanian
0419	Russian
044f	Sanskrit
081a	Serbian_Latin
0c1a	Serbian_Cyrillic
041b	Slovak
0424	Slovenian
040a	Spanish_Traditional_Sort
080a	Spanish_Mexican
0c0a	Spanish_Modern_Sort
100a	Spanish_Guatemala
140a	Spanish_Costa_Rica
180a	Spanish_Panama
1c0a	Spanish_Dominican_Republic
200a	Spanish_Venezuela

240a	Spanish_Colombia
280a	Spanish_Peru
2c0a	Spanish_Argentina
300a	Spanish_Ecuador
340a	Spanish_Chile
380a	Spanish_Uruguay
3c0a	Spanish_Paraguay
400a	Spanish_Bolivia
440a	Spanish_El_Salvador
480a	Spanish_Honduras
4c0a	Spanish_Nicaragua
500a	Spanish_Puerto_Rico
0441	Swahili
041d	Swedish
081d	Swedish_Finland
0449	Tamil
0444	Tatar
041e	Thai
041f	Turkish
0422	Ukrainian
0420	Urdu
0443	Uzbek_Latin

0843	Uzbek_Cyrillic
042a	Vietnamese

Example

```

MsgBox(0, "Your OS Language:", _Language())

Func _Language()
Select
    Case StringInStr("0413 0813", @OSLang)
        Return "Dutch"
    Case StringInStr("0409 0809 0c09 1009 1409 1809 1c09 2009 2409 2809 2c09 3009
3409", @OSLang)
        Return "English"
    Case StringInStr("040c 080c 0c0c 100c 140c 180c", @OSLang)
        Return "French"
    Case StringInStr("0407 0807 0c07 1007 1407", @OSLang)
        Return "German"
    Case StringInStr("0410 0810", @OSLang)
        Return "Italian"
    Case StringInStr("0414 0814", @OSLang)
        Return "Norwegian"
    Case StringInStr("0415", @OSLang)
        Return "Polish"
    Case StringInStr("0416 0816", @OSLang)
        Return "Portuguese"
    Case StringInStr("040a 080a 0c0a 100a 140a 180a 1c0a 200a 240a 280a 2c0a 300a
340a 380a 3c0a 400a 440a 480a 4c0a 500a", @OSLang)
        Return "Spanish"
    Case StringInStr("041d 081d", @OSLang)
        Return "Swedish"
    Case Else
        Return "Other (can't determine with @OSLang directly)"
EndSelect
EndFunc

```

Fuentes estándares de Windows

Lista basada en [Fuentes Estándares de Windows](#).

Las fuentes recomendadas son resaltadas en amarillo.

NOMBRE DE FUENTE	<u>Win2000</u>	<u>WinXP</u>	<u>VISTA</u>	<u>Win7</u>
Arial	x	x	x	x
Arial Black	x	x	x	x
Book Antiqua		x	x	
Calibri			x	x
Cambria			x	x
Candara			x	x
Comic Sans MS	x	x	x	x
Consolas			x	x
Constantia			x	x
Corbel			x	x
Courier	x	x	x	x
Courier New	x	x	x	x
Estrangelo Edessa		x	x	x
Franklin Gothic Medium		x	x	x
Gautami		x	x	x
Gabriola				x
Georgia	x	x	x	
Georgia Italic Impact		x		
Impact	x	x	x	x
Latha		x	x	x
Lucida Console	x	x	x	x

Lucida Sans Console		x	x	x
Lucida Sans Unicode	x	x	x	x
Marlett	x	x	x	x
Modern	x	x	x	x
Modern MS Sans Serif		x		
MS Sans Serif	x	x	x	x
MS Serif	x	x	x	x
MV Boli		x	x	x
Nyala			x	x
Palatino Linotype	x	x	x	x
Roman	x	x	x	x
Script	x	x	x	x
Sego Print			x	x
Sego Script			x	x
Segoe UI			x	x
Small Fonts	x	x	x	
Symbol	x	x	x	x
Tahoma	x	x	x	x
Tempus Sans ITC	x			
Times New Roman	x	x	x	x
Trebuchet	x	x	x	x
Tunga		x	x	x

Verdana	X	X	X	X
Webdings	X	X	X	X
Westminster		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				

Códigos de Mensajes de Windows

Ordenado por nombre

Ordenado por código

Mensaje	Código	Código	Mensaje
WM_ACTIVATE	0x0006	0x0000	WM_NULL
WM_ACTIVATEAPP	0x001C	0x0001	WM_CREATE
WM_AFXFIRST	0x0360	0x0002	WM_DESTROY
WM_AFXLAST	0x037F	0x0003	WM_MOVE
WM_APP	0x8000	0x0005	WM_SIZE
WM_APPCOMMAND	0x0319	0x0006	WM_ACTIVATE
WM_ASKCBFORMATNAME	0x030C	0x0007	WM_SETFOCUS

WM_CANCELJOURNAL	0x004B	0x0008	WM_KILLFOCUS
WM_CANCELMODE	0x001F	0x000A	WM_ENABLE
WM_CAPTURECHANGED	0x0215	0x000B	WM_SETREDRAW
WM_CHANGEBCBCHAIN	0x030D	0x000C	WM_SETTEXT
WM_CHANGEUISTATE	0x0127	0x000D	WM_GETTEXT
WM_CHAR	0x0102	0x000E	WM_GETTEXTLENGTH
WM_CHARTOITEM	0x002F	0x000F	WM_PAINT
WM_CHILDACTIVATE	0x0022	0x0010	WM_CLOSE
WM_CLEAR	0x0303	0x0011	WM_QUERYENDSESSION
WM_CLOSE	0x0010	0x0013	WM_QUERYOPEN
WM_COMMAND	0x0111	0x0016	WM_ENDSESSION
WM_COMMNOTIFY	0x0044	0x0012	WM_QUIT
WM_COMPACTING	0x0041	0x0014	WM_ERASEBGND
WM_COMPAREITEM	0x0039	0x0015	WM_SYSCOLORCHANGE
WM_CONTEXTMENU	0x007B	0x0018	WM_SHOWWINDOW
WM_COPY	0x0301	0x001A	WM_WININICHANGE
WM_COPYDATA	0x004A	0x001A	WM_SETTINGCHANGE
WM_CREATE	0x0001	0x001B	WM_DEVMODECHANGE
WM_CTLCOLORBTN	0x0135	0x001C	WM_ACTIVATEAPP
WM_CTLCOLORDLG	0x0136	0x001D	WM_FONTCHANGE
WM_CTLCOLOREDIT	0x0133	0x001E	WM_TIMECHANGE
WM_CTLCOLORLISTBOX	0x0134	0x001F	WM_CANCELMODE

WM_CTLCOLORMSGBOX	0x0132	0x0020	WM_SETCURSOR
WM_CTLCOLORSCROLLBAR	0x0137	0x0021	WM_MOUSEACTIVATE
WM_CTLCOLORSTATIC	0x0138	0x0022	WM_CHILDACTIVATE
WM_CUT	0x0300	0x0023	WM_QUEUESYNC
WM_DEADCHAR	0x0103	0x0024	WM_GETMINMAXINFO
WM_DELETEITEM	0x002D	0x0026	WM_PAINTICON
WM_DESTROY	0x0002	0x0027	WM_ICONERASEBKGND
WM_DESTROYCLIPBOARD	0x0307	0x0028	WM_NEXTDLGCTL
WM_DEVICECHANGE	0x0219	0x002A	WM_SPOOLERSTATUS
WM_DEVMODECHANGE	0x001B	0x002B	WM_DRAWITEM
WM_DISPLAYCHANGE	0x007E	0x002C	WM_MEASUREITEM
WM_DRAWCLIPBOARD	0x0308	0x002D	WM_DELETEITEM
WM_DRAWITEM	0x002B	0x002E	WM_VKEYTOITEM
WM_DROPFILES	0x0233	0x002F	WM_CHARTOITEM
WM_ENABLE	0x000A	0x0030	WM_SETFONT
WM_ENDSESSION	0x0016	0x0031	WM_GETFONT
WM_ENTERIDLE	0x0121	0x0032	WM_SETHOTKEY
WM_ENTERMENULOOP	0x0211	0x0033	WM_GETHOTKEY
WM_ENTERSIZEMOVE	0x0231	0x0037	WM_QUERYDRAGICON
WM_ERASEBKGND	0x0014	0x0039	WM_COMPAREITEM
WM_EXITMENULOOP	0x0212	0x003D	WM_GETOBJECT
WM_EXITSIZEMOVE	0x0232	0x0041	WM_COMPACTING

WM_FONTCHANGE	0x001D	0x0044	WM_COMMNOTIFY
WM_GETDLGCODE	0x0087	0x0046	WM_WINDOWPOSCHANGING
WM_GETFONT	0x0031	0x0047	WM_WINDOWPOSCHANGED
WM_GETHOTKEY	0x0033	0x0048	WM_POWER
WM_GETICON	0x007F	0x004A	WM_COPYDATA
WM_GETMINMAXINFO	0x0024	0x004B	WM_CANCELJOURNAL
WM_GETOBJECT	0x003D	0x004E	WM_NOTIFY
WM_GETTEXT	0x000D	0x0050	WM_INPUTLANGCHANGEREQUEST
WM_GETTEXTLENGTH	0x000E	0x0051	WM_INPUTLANGCHANGE
WM_HANDHELDFIRST	0x0358	0x0052	WM_TCARD
WM_HANDHELDLAST	0x035F	0x0053	WM_HELP
WM_HELP	0x0053	0x0054	WM_USERCHANGED
WM_HOTKEY	0x0312	0x0055	WM_NOTIFYFORMAT
WM_HSCROLL	0x0114	0x007B	WM_CONTEXTMENU
WM_HSCROLLCLIPBOARD	0x030E	0x007C	WM_STYLECHANGING
WM_ICONERASEBKGND	0x0027	0x007D	WM_STYLECHANGED
WM ime_CHAR	0x0286	0x007E	WM_DISPLAYCHANGE
WM ime_COMPOSITION	0x010F	0x007F	WM_GETICON
WM ime_COMPOSITIONFULL	0x0284	0x0080	WM_SETICON
WM ime_CONTROL	0x0283	0x0081	WM_NCCREATE
WM ime_ENDCOMPOSITION	0x010E	0x0082	WM_NCDESTROY
WM ime_KEYDOWN	0x0290	0x0083	WM_NCCALCSIZE

WM_IME_KEYLAST	0x010F	0x0084	WM_NCHITTEST
WM_IME_KEYUP	0x0291	0x0085	WM_NCPAINT
WM_IME_NOTIFY	0x0282	0x0086	WM_NCACTIVATE
WM_IME_REQUEST	0x0288	0x0087	WM_GETDLGCODE
WM_IME_SELECT	0x0285	0x0088	WM_SYNCPAINT
WM_IME_SETCONTEXT	0x0281	0x00A0	WM_NCMOUSEMOVE
WM_IME_STARTCOMPOSITION	0x010D	0x00A1	WM_NCLBUTTONDOWN
WM_INITDIALOG	0x0110	0x00A2	WM_NCLBUTTONUP
WM_INITMENU	0x0116	0x00A3	WM_NCLBUTTONDOWNDBLCLK
WM_INITMENUPOPUP	0x0117	0x00A4	WM_NCRBUTTONDOWN
WM_INPUT	0x00FF	0x00A5	WM_NCRBUTTONUP
WM_INPUTLANGCHANGE	0x0051	0x00A6	WM_NCRBUTTONDOWNDBLCLK
WM_INPUTLANGCHANGEREQUEST	0x0050	0x00A7	WM_NCMBUTTONDOWN
WM_KEYDOWN	0x0100	0x00A8	WM_NCMBUTTONUP
WM_KEYFIRST	0x0100	0x00A9	WM_NCMBUTTONDOWNDBLCLK
WM_KEYLAST	0x0108	0x00AB	WM_NCXBUTTONDOWN
WM_KEYLAST	0x0109	0x00AC	WM_NCXBUTTONUP
WM_KEYUP	0x0101	0x00AD	WM_NCXBUTTONDOWNDBLCLK
WM_KILLFOCUS	0x0008	0x00FF	WM_INPUT
WM_LBUTTONDOWNDBLCLK	0x0203	0x0100	WM_KEYFIRST
WM_LBUTTONDOWN	0x0201	0x0100	WM_KEYDOWN
WM_LBUTTONUP	0x0202	0x0101	WM_KEYUP

WM_MBUTTONDOWNDBLCLK	0x0209	0x0102	WM_CHAR
WM_MBUTTONDOWN	0x0207	0x0103	WM_DEADCHAR
WM_MBUTTONUP	0x0208	0x0104	WM_SYSKEYDOWN
WM_MDIActivate	0x0222	0x0105	WM_SYSKEYUP
WM_MDICASCADE	0x0227	0x0106	WM_SYSCHAR
WM_MDICREATE	0x0220	0x0107	WM_SYSDEADCHAR
WM_MDIDESTROY	0x0221	0x0109	WM_UNICHAR
WM_MDIGETACTIVE	0x0229	0x0109	WM_KEYLAST
WM_MDIICONARRANGE	0x0228	0x0108	WM_KEYLAST
WM_MDIMAXIMIZE	0x0225	0x010D	WM ime_startcomposition
WM_MDIEXT	0x0224	0x010E	WM ime_endcomposition
WM_MDIREFRESHMENU	0x0234	0x010F	WM ime_composition
WM_MDIRESTORE	0x0223	0x010F	WM ime_keylast
WM_MDISETMENU	0x0230	0x0110	WM_INITDIALOG
WM_MDTILE	0x0226	0x0111	WM_COMMAND
WM_MEASUREITEM	0x002C	0x0112	WM_SYSCOMMAND
WM_MENUCHAR	0x0120	0x0113	WM_TIMER
WM_MENUCOMMAND	0x0126	0x0114	WM_HSCROLL
WM_MENUDRAG	0x0123	0x0115	WM_VSCROLL
WM_MENUGETOBJECT	0x0124	0x0116	WM_INITMENU
WM_MENURBUTTONUP	0x0122	0x0117	WM_INITMENUPOPUP
WM_MENUSELECT	0x011F	0x011F	WM_MENUSELECT

WM_MOUSEACTIVATE	0x0021	0x0120	WM_MENUCHAR
WM_MOUSEFIRST	0x0200	0x0121	WM_ENTERIDLE
WM_MOUSEOVER	0x02A1	0x0122	WM_MENURBUTTONUP
WM_MOUSELAST(2K,XP,2k3)	0x020D	0x0123	WM_MENUDRAG
WM_MOUSELAST(95)	0x0209	0x0124	WM_MENUGETOBJECT
WM_MOUSELAST(NT4,98)	0x020A	0x0125	WM_UNINITMENUPOPUP
WM_MOUSELEAVE	0x02A3	0x0126	WM_MENUCOMMAND
WM_MOUSEMOVE	0x0200	0x0127	WM_CHANGEUISTATE
WM_MOUSEWHEEL	0x020A	0x0128	WM_UPDATEUISTATE
WM_MOVE	0x0003	0x0129	WM_QUERYUISTATE
WM_MOVING	0x0216	0x0132	WM_CTLCOLORMSGBOX
WM_NCACTIVATE	0x0086	0x0133	WM_CTLCOLOREDIT
WM_NCCALCSIZE	0x0083	0x0134	WM_CTLCOLORLISTBOX
WM_NCCREATE	0x0081	0x0135	WM_CTLCOLORBTN
WM_NCDESTROY	0x0082	0x0136	WM_CTLCOLORDLG
WM_NCHITTEST	0x0084	0x0137	WM_CTLCOLORSCROLLBAR
WM_NCLBUTTONDOWNDBLCLK	0x00A3	0x0138	WM_CTLCOLORSTATIC
WM_NCLBUTTONDOWN	0x00A1	0x0200	WM_MOUSEFIRST
WM_NCLBUTTONUP	0x00A2	0x0200	WM_MOUSEMOVE
WM_NCMBUTTONDOWNDBLCLK	0x00A9	0x0201	WM_LBUTTONDOWN
WM_NCMBUTTONDOWN	0x00A7	0x0202	WM_LBUTTONUP
WM_NCMBUTTONUP	0x00A8	0x0203	WM_LBUTTONDBLCLK

WM_NCMOUSEHOVER	0x02A0	0x0204	WM_RBUTTONDOWN
WM_NCMOUSELEAVE	0x02A2	0x0205	WM_RBUTTONUP
WM_NCMOUSEMOVE	0x00A0	0x0206	WM_RBUTTONDOWNDBLCLK
WM_NCPAINT	0x0085	0x0207	WM_MBUTTONDOWN
WM_NCRBUTTONDOWNDBLCLK	0x00A6	0x0208	WM_MBUTTONUP
WM_NCRBUTTONDOWN	0x00A4	0x0209	WM_MBUTTONDOWNDBLCLK
WM_NCRBUTTONUP	0x00A5	0x0209	WM_MOUSELAST(95)
WM_NCXBUTTONDOWNDBLCLK	0x00AD	0x020A	WM_MOUSEWHEEL
WM_NCXBUTTONDOWNDOWN	0x00AB	0x020A	WM_MOUSELAST(NT4,98)
WM_NCXBUTTONUP	0x00AC	0x020B	WM_XBUTTONDOWN
WM_NEXTDLGCTL	0x0028	0x020C	WM_XBUTTONUP
WM_NEXTMENU	0x0213	0x020D	WM_XBUTTONDOWNDBLCLK
WM_NOTIFY	0x004E	0x020D	WM_MOUSELAST(2K,XP,2k3)
WM_NOTIFYFORMAT	0x0055	0x0210	WM_PARENTNOTIFY
WM_NULL	0x0000	0x0211	WM_ENTERMENULOOP
WM_PAINT	0x000F	0x0212	WM_EXITMENULOOP
WM_PAINTCLIPBOARD	0x0309	0x0213	WM_NEXTMENU
WM_PAINTICON	0x0026	0x0214	WM_SIZING
WM_PALETTECHANGED	0x0311	0x0215	WM_CAPTURECHANGED
WM_PALETTEISCHANGING	0x0310	0x0216	WM_MOVING
WM_PARENTNOTIFY	0x0210	0x0218	WM_POWERBROADCAST
WM_PASTE	0x0302	0x0219	WM_DEVICECHANGE

WM_PENWINFIRST	0x0380	0x0220	WM_MDICREATE
WM_PENWINLAST	0x038F	0x0221	WM_MDIDESTROY
WM_POWER	0x0048	0x0222	WM_MDIACTIVATE
WM_POWERBROADCAST	0x0218	0x0223	WM_MDIRESTORE
WM_PRINT	0x0317	0x0224	WM_MDINEXT
WM_PRINTCLIENT	0x0318	0x0225	WM_MDIMAXIMIZE
WM_QUERYDRAGICON	0x0037	0x0226	WM_MDTILE
WM_QUERYENDSESSION	0x0011	0x0227	WM_MDICASCADE
WM_QUERYNEWPALETTE	0x030F	0x0228	WM_MDIICONARRANGE
WM_QUERYOPEN	0x0013	0x0229	WM_MDIGETACTIVE
WM_QUERYUISTATE	0x0129	0x0230	WM_MDISETMENU
WM_QUEUESYNC	0x0023	0x0231	WM_ENTERSIZEMOVE
WM_QUIT	0x0012	0x0232	WM_EXITSIZEMOVE
WM_RBUTTONDOWNDBLCLK	0x0206	0x0233	WM_DROPFILES
WM_RBUTTONDOWN	0x0204	0x0234	WM_MDIREFRESHMENU
WM_RBUTTONUP	0x0205	0x0281	WM_IME_SETCONTEXT
WM_RENDERALLFORMATS	0x0306	0x0282	WM_IME_NOTIFY
WM_RENDERFORMAT	0x0305	0x0283	WM_IME_CONTROL
WM_SETCURSOR	0x0020	0x0284	WM_IME_COMPOSITIONFULL
WM_SETFOCUS	0x0007	0x0285	WM_IME_SELECT
WM_SETFONT	0x0030	0x0286	WM_IME_CHAR
WM_SETHOTKEY	0x0032	0x0288	WM_IME_REQUEST

WM_SETICON	0x0080	0x0290	WM_IME_KEYDOWN
WM_SETREDRAW	0x000B	0x0291	WM_IME_KEYUP
WM_SETTEXT	0x000C	0x02A1	WM_MOUSEOVER
WM_SETTINGCHANGE	0x001A	0x02A3	WM_MOUSELEAVE
WM_SHOWWINDOW	0x0018	0x02A0	WM_NCMOUSEOVER
WM_SIZE	0x0005	0x02A2	WM_NCMOUSELEAVE
WM_SIZECLIPBOARD	0x030B	0x02B1	WM_WTSSESSION_CHANGE
WM_SIZING	0x0214	0x02C0	WM_TABLET_FIRST
WM_SPOOLERSTATUS	0x002A	0x02DF	WM_TABLET_LAST
WM_STYLECHANGED	0x007D	0x0300	WM_CUT
WM_STYLECHANGING	0x007C	0x0301	WM_COPY
WM_SYNCPAINT	0x0088	0x0302	WM_PASTE
WM_SYSCHAR	0x0106	0x0303	WM_CLEAR
WM_SYSCOLORCHANGE	0x0015	0x0304	WM_UNDO
WM_SYSCOMMAND	0x0112	0x0305	WM_RENDERFORMAT
WM_SYSDEADCHAR	0x0107	0x0306	WM_RENDERALLFORMATS
WM_SYSKEYDOWN	0x0104	0x0307	WM_DESTROYCLIPBOARD
WM_SYSKEYUP	0x0105	0x0308	WM_DRAWCLIPBOARD
WM_TABLET_FIRST	0x02C0	0x0309	WM_PAINTCLIPBOARD
WM_TABLET_LAST	0x02DF	0x030A	WM_VSCROLLCLIPBOARD
WM_TCARD	0x0052	0x030B	WM_SIZECLIPBOARD
WM_THEMECHANGED	0x031A	0x030C	WM_ASKCBFORMATNAME

WM_TIMECHANGE	0x001E	0x030D	WM_CHANGEBCCHAIN
WM_TIMER	0x0113	0x030E	WM_HSCROLLCLIPBOARD
WM_UNDO	0x0304	0x030F	WM_QUERYNEWPALETTE
WM_UNICHAR	0x0109	0x0310	WM_PALETTEISCHANGING
WM_UNINITMENUPOPUP	0x0125	0x0311	WM_PALETTECHANGED
WM_UPDATEUISTATE	0x0128	0x0312	WM_HOTKEY
WM_USER	0x0400	0x0317	WM_PRINT
WM_USERCHANGED	0x0054	0x0318	WM_PRINTCLIENT
WM_VKEYTOITEM	0x002E	0x0319	WM_APPCOMMAND
WM_VSCROLL	0x0115	0x031A	WM_THEMECHANGED
WM_VSCROLLCLIPBOARD	0x030A	0x0358	WM_HANDHELDFIRST
WM_WINDOWPOSCHANGED	0x0047	0x035F	WM_HANDHELDLAST
WM_WINDOWPOSCHANGING	0x0046	0x0360	WM_AFXFIRST
WM_WININICHANGE	0x001A	0x037F	WM_AFXLAST
WM_WTSSESSION_CHANGE	0x02B1	0x0380	WM_PENWINFIRST
WM_XBUTTONDOWNDBLCLK	0x020D	0x038F	WM_PENWINLAST
WM_XBUTTONDOWN	0x020B	0x0400	WM_USER
WM_XBUTTONUP	0x020C	0x8000	WM_APP

