**AutoIt v.3 Nachschlagewerk**

**Von Hingo**



Maurice Maréchal

## Inhaltsverzeichnis


Was ist AutoIt ..... 3

Variablen ..... 3

- Aufbau ..... 4
- Globale Variablen ..... 4
- Lokale Variablen ..... 5
- Dim Variable ..... 5
- Warum Variablen deklarieren? ..... 5

Arrays ..... 6

- Aufbau ..... 6

Schleifen ..... 8

- While ..... 8
- Do ..... 8
- For ..... 9

Bedingungen ..... 10

- If ..... 10
- Else ..... 11
- ElseIf ..... 11

Konstanten ..... 12

- Globale Konstante ..... 12
- Lokale Konstante ..... 12

Kommentare ..... 13

Unterschied Syntax und Semantik ..... 14

- Syntax ..... 14
- Semantik ..... 14

GUI ..... 15

- GUICreate ..... 15
- GUICtrlCreateButton ..... 15
- GUISetState ..... 15
- GUICtrlCreateCombo ..... 16
- GUICtrlCreateDate ..... 16
- GUICtrlCreateMenu ..... 17
- GUICtrlCreateLabel ..... 18
- GUICtrlCreateInput ..... 19
- GUICtrlCreateIcon ..... 20

GUICtrlCreateCheckbox ..... 20
GUISetBkColor ..... 21
Obj-Funktionen ..... 22
ObjGet ..... 22
ObjCreate ..... 22
ObjName ..... 23
RUN ..... 24
Dir-Funktion ..... 25
DirCreate ..... 25
DirCopy ..... 25
DirGetSize ..... 25
DirMove ..... 25
DirRemove ..... 26
MsgBox ..... 27
File Funktionen ..... 28
FileGetSize ..... 28
FileMove ..... 28
FileOpen ..... 29
FileOpenDialog ..... 30
FileSelectFolder ..... 31
FileSaveDialog ..... 31
FileWrite ..... 32
FileClose ..... 32
FileCopy ..... 32
FileDelete ..... 33
FileExists ..... 33
FileRecycle ..... 33
Versionsnummer ..... 35

## Was ist AutoIt

AutoIt ist eine Freeware-Skriptsprache. Sie wurde entwickelt, um die Windows-GUI (Grafische Benutzeroberfläche) zu automatisieren. Sie nutzt eine Kombination von simulierten Tastendrücken, Mausbewegungen und -klicks sowie Windows- bzw. Control-Manipulation, um Aufgaben zu automatisieren, die in anderen Sprachen nicht möglich sind oder nur mit befriedigender Stabilität und Zuverlässigkeit möglich wären (z.B. VBScript). AutoIt ist auch sehr klein, unabhängig und läuft auf allen Versionen von Windows.

Die Features von AutoIt:

- Einfach zu lernende BASIC-ähnliche Syntax
- Tastendrücke und Mausbewegungen simulieren
- Windows und Prozesse manipulieren
- Mit allen Windows-Standardsteuerelementen arbeiten
- Skripte können in direkt ausführbare exe-Dateien umgewandelt werden
- Grafische Benutzeroberflächen (GUIs) erstellen
- COM Unterstützung
- Reguläre Ausdrücke
- Direkt externe DLLs und Windows-API-Funktionen aufrufen
- Programmierbare RunAs-Funktionen
- Detaillierte Hilfedatei und große communitybasierende Supportforen
- Kompatibel mit Windows 2000 / XP / 2003 / Vista / 2008 / 7
- Unicode- und x64-Support
- Digital signiert für sorgenfreies Arbeiten
- Funktioniert mit Windows Vista's Benutzerkontensteuerung (UAC)
- Skripte mit Aut2Exe in selbstständig ausführbare Dateien kompilieren

## Variablen

Variablen sind unser Speichermedium in denen wir Daten abspeichern können um sie zu einem Späteren Zeitpunkt z.B für eine Berechnung wieder zu verwenden.

Wenn Variablen in einer MsgBox oder GUICtrlCreateInput ausgegeben werden sollen, werden sie ohne Anführungszeichen geschrieben.

Beispiel:

```
Global $anfang = GUICtrlCreateInput ( $variable, 20, 35, 300,20)

MsgBox($MB_SYSTEMMODAL, “Titel“ , $variable,)
```

### Aufbau

Das erste was wir angeben können ist der Bereich in der die Variable gültig ist. Dieser kann Global sein (überall im Script verfügbar), Lokal (nur innerhalb einer Funktion) oder Dim (Ist nur dann Lokal wenn es keine Globale Variable gleichen Namens gibt). Danach kommt ein **"$"** das der Variable vorangestellt wird und danach der Name. Zuletzt wird mit einem **"="** der Inhalt der Variable festgelegt.

Beispiele:

```
Local $wetter1 = sonnig

Global $wetter2 = regnerisch

Dim $wetter3 = bewölkt
```

### Globale Variablen

Globale Variablen können überall aus dem Skript angesprochen werden. Zum deklarieren steht uns das Schlüsselwort „Global“ zur Verfügung. Eine globale Variable muss nur einmal im ganzen Skript als global markiert sein. Danach steht diese ständig zur Verfügung.

Beispiel:

```
1 Global $var
2
3 ; Die Funktion IsDeclared zeigt uns an ob eine Variable deklariert wurde.
4 ; Der Rückgabewert der Funktion beträgt für globale Variablen gleich 1.
5 MsgBox(0, '', IsDeclared('var'))
6
7 ; Da wir der Variable noch keinen Wert zugewiesen haben ist diese leer.
8 MsgBox(0, '', $var)
```

**Lokale Variablen**

Lokale Variablen sind nicht überall im Skript ansprechbar. Allerdings gilt diese Regelung nur, wenn die Variable in einer eigenen Funktion lokal deklariert wird. Zum deklarieren steht uns in AutoIt das Schlüsselwort „Local“ zur Verfügung.

Beispiel:

```
1 Func EigeneFunktion()
2       Local $LokaleVariable = 'Text'
3
4       ; Der Rückgabewert -1 zeigt uns an, dass die Variable nur lokal zur
5       Verfügung steht.
6       MsgBox(0, 'Eigene Funktion - IsDeclared',
        IsDeclared('LokaleVariable'))
7
8
9       MsgBox(0, 'Eigene Funktion - Lokale Variable', $LokaleVariable)
10 EndFunc
```

Wenn nun eine neue “Func“ gestartet wird müsste die Variable “$LokaleVariable“ nochmals deklariert werden.

**Dim Variable**

Wenn eine Variable nur in einer Funktion ist wird sie automatisch “Local“ wenn sie in 2 oder mehr Funktionen vorhanden ist werden sie als “Global“ deklariert.

**Warum Variablen deklarieren?**

Variablen zu deklarieren ist sinnvoll, da dadurch Fehler umgangen werden. Denn ohne eine Deklaration, also ohne eine Zuweisung weis das Programm nicht ob es eine Variable in dem gesamten Skript oder nur in einer Funktion zugreifen soll.

## Arrays

Da wir pro Variable nur ein Inhalt festlegen können, gibt es Arrays diese sind fast das gleiche wie Variablen nur das man in dieser mehrere Inhalte speichern kann.

## Aufbau

Wie speichert man aber Daten in einem Array ab? Das Ganze ist exakt wie bei einer Variable bis auf das wir als erstes den Array Umfang festlegen und beim Speichern von Daten die Zeile angeben in die der Datensatz gespeichert wird.
Die Anzahl der Elemente wird in eckigen Klammern hinter dem Variablennamen angegeben.

Bevor ich ein Array mit Daten füllen kann, muss ich es deklarieren, also einen Variablennamen vergeben und optional die Anzahl der Elemente festlegen.
Es empfiehlt sich an den Beginn des Variablennamens 'a' oder 'ar' zu setzen um immer sofort zu wissen, dass dieses ein Array ist.
Jedes Element in einem Array wird über seinen Index angesprochen. Das ist gewissermaßen die Hausnummer des Elements.
Hierbei ist zu beachten, dass im Array der erste Index = 0 ist.

Beispiel:

```
Local $aTest[0] = 'Wert 1'

Global $aTest[1] = 'Wert 2'

Dim $aTest[2] = 'Wert 3'

$aTest[3] = 'Wert 4'
```

Beispiel:
Mit dem folgenden Beispiel wird gezeigt, wie man Programmieraufwand durch Einsatz eines Array verringern kann.

Eine GUI mit Controls wird deklariert. Die Controls sollen per Button wahlweise aktiv/deaktiv sein. Die Art der Controls wird offen gelassen.

| Variante ohne Array | Variante mit Array |
|---|---|
| `Local $aktiv = True` | `Local $aktiv = True` |
| `$GUI = GUICreate (“Titel”)` | `Local $arControl [8]` |
| `$control1 = GUIctrlCreate` | `$GUI = GUICreate (“Titel”)` |
| `$control2 = GUIctrlCreate` | `$arControl [0] = GuiCtrlCreate` |
| `$control3 = GUIctrlCreate` | `$arControl [1] = GuiCtrlCreate` |
| `$control4 = GUIctrlCreate` | `$arControl [2] = GuiCtrlCreate` |
| `$control5 = GUIctrlCreate` | `$arControl [3] = GuiCtrlCreate` |
| `$control6 = GUIctrlCreate` | `$arControl [4] = GuiCtrlCreate` |

```
$control7 = GUIctrlCreate

$button = GUICtrlCreate

If $msg = $button Then _SwitchActive
()

Func _SwitchActive ()

      Local $state

      If $aktiv Then

            $state = GUI_DISABLE

      Else

            $state = $GUI_ENABLE

      EndIf

      $aktiv = Not aktiv

      GUICtrlState ($control1, $state)

      GUICtrlState ($control2, $state)

      GUICtrlState ($control3, $state)

      GUICtrlState ($control4, $state)

      GUICtrlState ($control5, $state)

      GUICtrlState ($control6, $state)

      GUICtrlState ($control7, $state)

EndFunc
```

```
$arControl [5] = GuiCtrlCreate

$arControl [6] = GuiCtrlCreate

$arControl [7] = GuiCtrlCreate

If $msg = $arControl [7] Then
_SwitchActive ()

Func _SwitchActive ()

      Local $state = $aktiv ?
$GUI_ENEBLE : $GUI_DISABLE

      $aktiv = Not $aktiv

      For $i = 0 To 6

            GUICtrlSetState
($arControl [$i], $state)

      Next

EndFunc
```

## Schleifen

Schleifen benötigt man für Funktionen die man öfters wiederholen will.

Darunter gibt es die While und Do Schleife die unendlich wiederholt werden solange der Wert “True“ ist. Dagegen kann man bei der For Schleife angeben wie viele Wiederholungen es geben soll.

### While

Eine While Schleife wird durch das Schlüsselwort “While“ eingeleitet und mit “WEnd“ beendet.

Die "While" Schleife wird solang ausgeführt solange der Wert neben "While" richtig ist.

Beispiel :

```
func _stop()
   Exit
endFunc
   HotKeySet("^{F9}","_stop")
While True
   Msgbox(64,"Ich...","...nerve dich bis Strg+F9")
WEnd
```

Eine While-Schleife wird ja bekanntlich so lange ausgeführt, wie die Bedingung "wahr" ist. Da das Schlüsselwort "true" immer wahr ist, wird die Schleife ewig ausgeführt.

**Wichtig** ist hier das erst geprüft wird ob der wert Richtig ist und dann die While Schleife angefangen wird.

### Do

Bei der Do Schleife wird erst der Inhalt ausgeführt und dann geprüft ob der Wert richtig ist, wenn der Wert richtig ist wird die Schleife von neuem ausgeführt.

Die Do-Schleife wird mit dem Schlüsselwort “Do“ eingeleitet und hört mit “Until“ auf.

Beispiel:

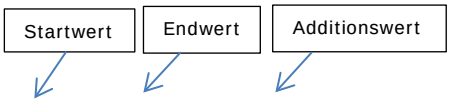
```
#include <MsgBoxConstants.au3>

Local $i = 0
Do
    MsgBox($MB_SYSTEMMODAL, "", "The value of $i is: " & $i) ; Display the value of $i.
    $i = $i + 1 ; Or $i += 1 can be used as well.
Until $i = 10 ; Increase the value of $i until it equals the value of 10.
```

**For**

Die For Schleife kann man eine bestimme Anzahl an Runden laufen lassen. Für diese benötigen wir eine Variable die den Startwert beinhaltet, einen End Wert und in welchen Schritten die Schleife Läuft.

Da bei jedem Durchlauf $i um eins erhöht wird, wissen wir immer in welcher Runde sich die Schleife befindet. In diesem Fall wird die Variable aber nicht um eins erhöht sondern um 10 das heißt, da der Startwert 0 ist und der Endwert 100, das wir 10 Durchläufe haben.

Beispiel:




```
For $i = 0 To 100 Step 10

    MsgBox($MB_SYSTEMMODAL, "", "Count down!" & @CRLF & $i)

Next

MsgBox($MB_SYSTEMMODAL, "", "Blast Off!")
```

Die For Schleife wird mit dem Schlüsselwort “For“ eingeleitet und mit dem Schlüsselwort “Next“ beendet.

## Bedingungen

### If

Zum Beispiel bei einer Passwortabfrage überprüft die If-Bedingung ob das richtige Passwort eingegeben wurde.

Beispiel:

```
$PW = “1234“

$Test_PW = InputBox (“Passwort“, “Bitte geben sie ein Passwort ein”)


If $PW = $Test_PW Then

      MSGBOX (0, “Passwort“, “Das Passwort ist richtig“)

EndIf
```

Als erstes werden die Variablen festgelegt. Die Variable “$PW“ steht dafür wie das Passwort heißen muss, die Variable “$Test_PW“ steht für das eingegebene Passwort.

**Else**

Wenn das Skript nun ausgeführt wird, passiert nichts wenn das Passwort falsch ist. Das heißt wir müssen noch eine else Funktion einbauen, die bewirkt das wenn das Passwort falsch ist das noch etwas passiert, also z.B. das eine MsgBox erscheint, mit dem Inhalt “Das Passwort ist falsch“.

Beispiel:

```
$PW = “1234“

$Test_PW = InputBox (“Passwort“, “Bitte geben sie ein Passwort ein“)


If $PW = $Test_PW Then

      MsgBox (0, “Passwort ist richtig“)

Else

      MsgBox (0, “Passwort“, “Das Passwort ist falsch“)

EndIf
```

**ElseIf**

Wenn wir eine zweite Abfrage starten wollen, wenn die erste Abfrage nicht korrekt war, können wir dies mit ElseIf tun.

```
$PW = “1234“

$Test_PW = InputBox (“Passwort“, “Bitte geben sie ein Passwort ein“)


If $PW = $Test_PW Then

      MsgBox (0, “Passwort”, “Passwort ist richtig“)

ElseIf $Test_PW = “1234”

      MsgBox (0, “Passwort“, “Passwort ist richtig“)

Else

      MsgBox (0, „Passwort“, „Passwort ist falsch“)

EndIf
```

## Konstanten

Im Gegensatz zu Variablen, können sich konstante Werte während ihrer gesamten Lebensdauer nicht ändern. Dies kann dann sinnvoll sein, wenn Konstanten am Anfang des Programms definiert werden, um sie dann nur an einer Stelle im Quellcode anpassen zu müssen.

Eine Konstante wird, mit dem Schlüsselwort “Const“ deklariert.

Konstanten müssen nur einmal im Skript geändert werden.

Beispiel:

Die Mehrwertsteuer. Wird sie erhöht oder gesenkt, so muss sie nur an einer Stelle des Programms geändert werden.

Beispiel:

```
Const $var = 123
```

## Globale Konstante

Globale Konstanten verhalten sich genauso wie globale Variablen. Der einzige Unterschied jedoch ist, dass deren Inhalt nicht verändert werden kann. Eine globale Konstante existiert bis zum Programmende. Bis dahin kann keine neue Variable mit dem gleichen Namen deklariert werden.

Beispiel:

```
Global Const $var = 123

Local $var = 777

MsgBox (0, “”, $var)
```

Dies würde eine Fehlermeldung erzeugen da die Konstante $var in dem Programmcode verändert wurde.

## Lokale Konstante

Lokale Konstanten verhalten sich genauso wie lokale Variablen. Der einzige Unterschied ist, dass der Inhalt nicht verändert werden kann solange die Konstante existiert.

Beispiel:

```
Func MyFunc ()
      ;Random erzeugt eine zufällige Zahl
      Local Const $var = Random ()
      $var = Random ()
      MsgBox (0, “”, $var)
EndFunc
```

## Kommentare

Kommentare sind nützlich um die einzelnen Funktionen des Programmcodes zu beschrieben ohne, dass sie von dem Programm gelesen werden.

Beispiel Aufbau:

Bei einer Zeile

```
;Kommentar
```

Bei mehreren Zeilen

```
#cs

      Das

      ist

      ein

      Kommentar

#ce
```

Beispiel Praxis:

```
If $PW = $Test_PW Then              ; überprüft ob $PW gleich $Test_PW ist.

#cs

Öffnet ein

Fenster mit dem

Inhalt

Passwort ist richtig

#ce

      MsgBox (0, “Passwort”, “Passwort ist richtig“)
```

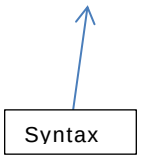
## Unterschied Syntax und Semantik

### Syntax

Die Syntax bezieht sich auf die Form und die Struktur von Zeichen, ohne auf die Bedeutung der verwendeten Zeichen einzugehen.

Beispiel:

```
If $PW = $Test_PW Then

      BoxMsg (0, “Passwort”, “Passwort ist richtig“)
```




### Semantik

Die Semantik befasst sich mit der Bedeutung syntaktisch korrekter Zeichenfolgen. Im Zusammenhang mit Programmiersprachen bedeutet Semantik der Sinn hinter dem Programmcode.

## GUI

Eine GUI ist eine grafische Oberfläche die es dem Anwender
ermöglicht den Programmablauf über Controls (Button, usw.…)
zu steuern.
Der Unterschied zu der Message Box ist, das eine GUI ermöglicht mehrere Werte gleichzeitig einzugeben.

## GUICreate

Die Funktion GUICreate öffnet ein Dialogfenster.

Aufbau:

```
GUICreate ("title" [, width [, height [, left = -1 [, top = -1 [, style = -
1 [, exStyle = -1 [, parent = 0]]]]]] )
```

Beispiel:

```
Local $hGUI = GUICreate("Example", 400, 100)
```

## GUICtrlCreateButton

Die Funktion erstellt einen Button.

Aufbau:

```
GUICtrlCreateButton ("text", left, top [, width [, height [, style = -1 [,
exStyle = -1]]]] )
```

Beispiel:

```
Local $idClose = GUICtrlCreateButton("Close", 210, 170, 85, 25)
```

## GUISetState

Die GUISetState ist dazu da, dass das Dialogfenster geöffnet wird.

Aufbau:

```
GUISetState ( [flag [, winhandle]] )
```

Beispiel:

```
GUICreate("My GUI")

    GUISetState(@SW_SHOW)
```

### GUICtrlCreateCombo

Die GUICtrlCreateCombo erstellt ein Drop-Down Menü.

Aufbau:

```
GUICtrlCreateCombo ("text", left, top [, width [, height [, style [,
exStyle]]]] )
```

Beispiel:

```
GUICtrlCreateCombo("item1", 10, 10) ; Erstellt das erste Item

GUICtrlSetData(-1, "item2|item3", "item3") ; Fügt andere Items hinzu und
                                             setzt einen neuen Standard
```

### GUICtrlCreateDate

Erstellt ein Auswahlmenü für ein Datum.

Aufbau:

```
GUICtrlCreateDate ("text", left, top [, width [, height [, style [,
exStyle]]]] )
```

Beispiel:

```
Func Example()
    Local $n, $msg

    GUICreate("Datum", 200, 200, 800, 200)
    $n = GUICtrlCreateDate("", 10, 10, 100, 20, $DTS_SHORTDATEFORMAT)
    GUISetState()

;Die Schleife wiederholt sich, bis der Benutzer eine Beenden-Aktion
auslöst
    Do
        $msg = GUIGetMsg()
    Until $msg = $GUI_EVENT_CLOSE


EndFunc
```

**GUICtrlCreateMenu**

Die Funktion erstellt eine Menüleiste für die bestehende GUI.

Aufbau:

```
GUICtrlCreateMenu (“submenutext“)
```

Beispiel:

```
Func Example()
    Local $defaultstatus = "Bereit", $filemenu, $fileitem
    Local $helpmenu, $infoitem, $exititem, $recentfilesmenu
    Local $viewmenu, $viewstatusitem, $cancelbutton
    Local $statuslabel, $msg, $file

    GUICreate("Mein GUI Menü", 300, 200)

    $filemenu = GUICtrlCreateMenu("&Datei")
    $fileitem = GUICtrlCreateMenuitem("Öffnen", $filemenu)
    GUICtrlSetState(-1, $GUI_DEFBUTTON)
    $helpmenu = GUICtrlCreateMenu("?")
    GUICtrlCreateMenuitem("Speichern", $filemenu)
    GUICtrlSetState(-1, $GUI_DISABLE)
    $infoitem = GUICtrlCreateMenu("Info", $helpmenu)
    $exititem = GUICtrlCreateMenuitem("Beenden", $filemenu)
    $recentfilesmenu = GUICtrlCreateMenu("Letzte Dateien", $filemenu, 1)

    GUICtrlCreateMenuitem("", $filemenu, 2)
    $viewmenu = GUICtrlCreateMenu("Ansicht", -1, 1)
    $viewstatusitem = GUICtrlCreateMenuitem("Statusbar", $viewmenu)
    GUICtrlSetState(-1, $GUI_CHECKED)
    GUICtrlCreateButton("OK", 50, 130, 70, 20)
    GUICtrlSetState(-1, $GUI_FOCUS)
    $cancelbutton = GUICtrlCreateButton("Abbrechen", 180, 130, 70, 20)

    $statuslabel = GUICtrlCreateLabel($defaultstatus, 0, 165, 300, 16,
    BitOR($SS_SIMPLE, $SS_SUNKEN))

    GUISetState()
    While 1
        $msg = GUIGetMsg()

        If $msg = $fileitem Then

            If @error <> 1 Then GUICtrlCreateMenuitem($file,
            $recentfilesmenu)
        EndIf
        If $msg = $viewstatusitem Then
        If BitAND(GUICtrlRead($viewstatusitem), $GUI_CHECKED) =
        $GUI_CHECKED Then
                GUICtrlSetState($viewstatusitem, $GUI_UNCHECKED)
                GUICtrlSetState($statuslabel, $GUI_HIDE)
            Else
                GUICtrlSetState($viewstatusitem, $GUI_CHECKED)
                GUICtrlSetState($statuslabel, $GUI_SHOW)
            EndIf
        EndIf
        If $msg = $GUI_EVENT_CLOSE Or $msg = $cancelbutton Or $msg =
        $exititem Then ExitLoop
        If $msg = $infoitem Then MsgBox(0, "Info", "Nur ein Test...")
    WEnd
```

**GUICtrlCreateLabel**
Erstellt ein Label in der GUI.

Aufbau:

```
GUICtrlCreateLabel (“text“, left, top)
```

Beispiel:

```
Func Example()
    Local $widthCell, $msg, $iOldOpt

    GUICreate("Mein GUI")

    $widthCell = 80
    GUICtrlCreateLabel("Test Text", 40, 30, $widthCell)

    GUISetState()
 Do
        $msg = GUIGetMsg()
    Until $msg = $GUI_EVENT_CLOSE

    $iOldOpt = Opt("GUICoordMode", $iOldOpt)
EndFunc
```

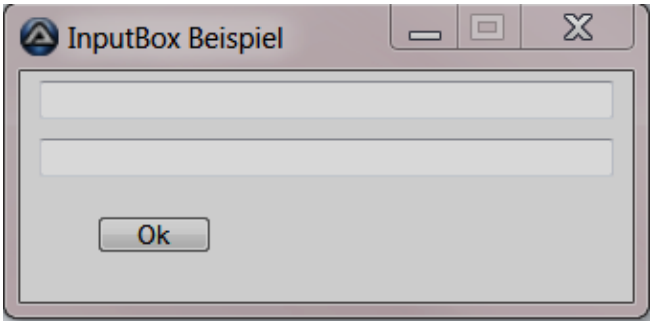
**GUICtrlCreateInput**
Erstellt eine Eingabe Dialogbox in der GUI.

Aufbau:

```
GUICtrlCreateInput (“text, left, top”)
```

Beispiel:

```
Func Example()
    Local $file, $btn, $msg

 GUICreate("Meine GUI die Dateien akzeptiert", 320, 120, @DesktopWidth / 2
 - 160, @DesktopHeight / 2 - 45, -1, 0x00000018)
 $file = GUICtrlCreateInput("", 10, 5, 300, 20)
 GUICtrlSetState(-1, $GUI_DROPACCEPTED)
 GUICtrlCreateInput("", 10, 35, 300, 20)
 $btn = GUICtrlCreateButton("Ok", 40, 75, 60, 20)

 GUISetState()

 $msg = 0
 While $msg <> $GUI_EVENT_CLOSE
     $msg = GUIGetMsg()
     Select
        Case $msg = $btn
           ExitLoop
     EndSelect
 WEnd

 MsgBox(4096, "Drag&Drop Datei", GUICtrlRead($file))
EndFunc
```



**GUICtrlCreateIcon**
Erstellt ein Icon Control in der GUI.

Aufbau:

```
GUICtrlCreateIcon (filename, iconname, left, top)
```

Beispiel:

```
#include <GUIConstantsEx.au3>

Example1()

Func Example1()
    GUICreate("Meine GUI Icons", 250, 250)

    GUICtrlCreateIcon("shell32.dll", 10, 20, 20)
    GUICtrlCreateIcon(@WindowsDir & "\cursors\horse.ani", -1, 20, 40, 32,
32)
    GUICtrlCreateIcon("shell32.dll", 7, 20, 75, 32, 32)
    GUISetState()

    While 1
        Local $msg = GUIGetMsg()

        If $msg = $GUI_EVENT_CLOSE Then ExitLoop
    WEnd
    GUIDelete()
EndFunc
```

**GUICtrlCreateCheckbox**

Erstellt eine Checkbox in der der GUI.

Aufbau:

```
GUICtrlCreateCheckbox (“text“, left, top)
```

Beispiel:

```
#include <GUIConstantsEx.au3>
Example()

Func Example()
    Local $msg
    GUICreate("Checkbox", 220, 50)

    GUICtrlCreateCheckbox("Checkbox 1", 10, 10, 120, 20)

    GUISetState()

    While 1
        $msg = GUIGetMsg()

        If $msg = $GUI_EVENT_CLOSE Then ExitLoop
    WEnd
EndFunc
```

**GUISetBkColor**
Ändert die Hintergrundfarbe des GUI-Fensters.

Aufbau:

```
GUISetBkColor(background-code)
```

Beispiel:

```
#include <GUIConstantsEx.au3>

Example()

Func Example()
    Local $msg

    GUICreate("Meine GUI")

    GUISetBkColor(0x996600)
    GUISetState()

    While 1
        $msg = GUIGetMsg()

        If $msg = $GUI_EVENT_CLOSE Then ExitLoop
    WEnd
EndFunc
```

Farbcode Tabelle

## Obj-Funktionen

### ObjGet

Ruft eine Referenz zu einem COM-Objekt ab, von einem vorhandenem Prozess oder Dateinamen.

Aufbau:

```
ObjGet ("filename" [, "classname"] )
```

Beispiel:

```
Local $oExcel = ObjGet("", "Excel.Application")
If @error Then
    MsgBox(0, "ExcelTest", "Fehler beim Referenzieren eines bestehenden
Excel Objekts. Fehlernummer: " & Hex(@error, 8))
    Exit
EndIf
```

### ObjCreate

Die Funktion ObjCreate startet verschiedene Programme in verschiedenen Instanzen zum Beispiel Excel oder Word.

Ein anderes Beispiel ist Internet Explorer, dort kann dann ausgewählt werden mit welchem Username und welches Passwort die Instanz geöffnet werden soll.

Beisspiel:

```
Local $oExcel = ObjCreate("Excel.Application")

$oExcel.Visible = 1

$oExcel.WorkBooks.Add

$oExcel.ActiveWorkBook.ActiveSheet.Cells(1, 1).Value = "Text" ; Fill a cell

$oExcel.ActiveWorkBook.Saved = 1
```

**ObjName**
Gibt den Namen oder die Schnittstelleninformation eines Objekts zurück.

Aufbau:

```
ObjName ( $Objectvariable [,Flag] )
```

Beispiel:

```
MsgBox(0, "", "Interface-Name von $oInternet ist: " & ObjName($oInternet
```

## RUN

Die Funktion Run startet ein externes Programm.

Aufbau:

```
Run ("program" [, "workingdir" [, show_flag [, opt_flag]]] )
```

Beispiel:

```
Run(@WindowsDir & "\Notepad.exe", "", @SW_MAXIMIZE)
```

## Dir-Funktion

Die Dir-Funktionen erstellen, kopieren, verschieben und löschen Ordner und Dateien.

### DirCreate

Erstellt ein Ordner.

Aufbau:

```
DirCreate ("path")
```

Beispiel:

```
DirCreate (“C:\Test\“)
```

### DirCopy

Kopiert einen Ordner mit Unterordner und Dateien.

Aufbau:

```
DirCopy ("source dir", "dest dir" [, flag] )
```

Beispiel:

```
DirCopy(@MyDocumentsDir, "C:\Sicherungen\Eigene Dateien", 1)
```

### DirGetSize

Gibt die Größe eines Ordners in Bytes an.

Aufbau:

```
DirGetSize ("path" [, flag] )
```

Beispiel:

```
Local $size = DirGetSize(@HomeDrive)
MsgBox(0, "", "Größe (MegaBytes):" & Round($size / 1024 / 1024))
```

### DirMove

Verschiebt Ordner mit allen Unterordnern und Dateien.

Aufbau:

```
DirMove ("source dir", "dest dir" [, flag] )
```

Beispiel:

```
DirMove ("C:\Test", "C:\Test1” )
```

**DirRemove**
Mit der Funktion DirRemove werden Ordner gelöscht.

Aufbau:

```
DirRemove ("path" [, recurse] )
```

Beispiel:

```
DirRemove ("C:\Test")
```

## MsgBox

Die MessageBox öffnet ein Windows Anzeigefenster.

Aufbau:

```
MsgBox (flag, "title", "text" [, timeout [, hwnd]] )
```

Beispiel:

```
MsgBox($MB_SYSTEMMODAL, "Test", "Diese Box wird sich in 10 Sekunden selbst
schließen", 10)
```

## File Funktionen

Die File Funktionen sind im Grunde dazu da, um zum Beispiel Textfiles zu erstellen, Textfiles zu verschieben oder um Dateigrößen auszugeben.

### FileGetSize

Gibt die Dateigröße aus.

Aufbau:

```
FileGetSize (“Filename“)
```

Beispiel:

```
Local $size = FileGetSize (“AutoIt.exe“)
```

### FileMove

Verschiebt die angegehbene/n Datei/en.

Aufbau:

```
FileMove (“source“, “dest“ [, flag] )
```

Beispiel:

```
FileMove (“C:\AutoIt.exe“, “D:\AutoIt.exe”)

#cs
Zweites Beispiel:
benutzt Flags '1' (überschreiben) und '8' (automatisch die
Zielverzeichnisstruktur erstellen) zusammen
verschiebt alle .txt-Dateien vom Temp-Verzeichnis in den Ordner TxtFiles
und überprüft, ob
Zielverzeichnisstruktur bereits existiert, falls nicht wird diese
automatisch erstellt
#ce

FileMove(@TempDir & "\*.txt", @TempDir & "\TxtFiles\", 9)
```

Das heißt, wenn der Flag 1 benutzt wird, werden die bestehenden Dateien überschrieben. Bei dem Flag 8 wird das Verzeichnis erstellt, wenn diese noch nicht existiert.

**FileOpen**
Öffnet eine Textdatei, die gelesen und beschrieben werden kann.

Aufbau:

```
FileOpen (“Filename“ [, mode])
```

Beispiel:

```
Local $file = FileOpen (“test.txt)
```

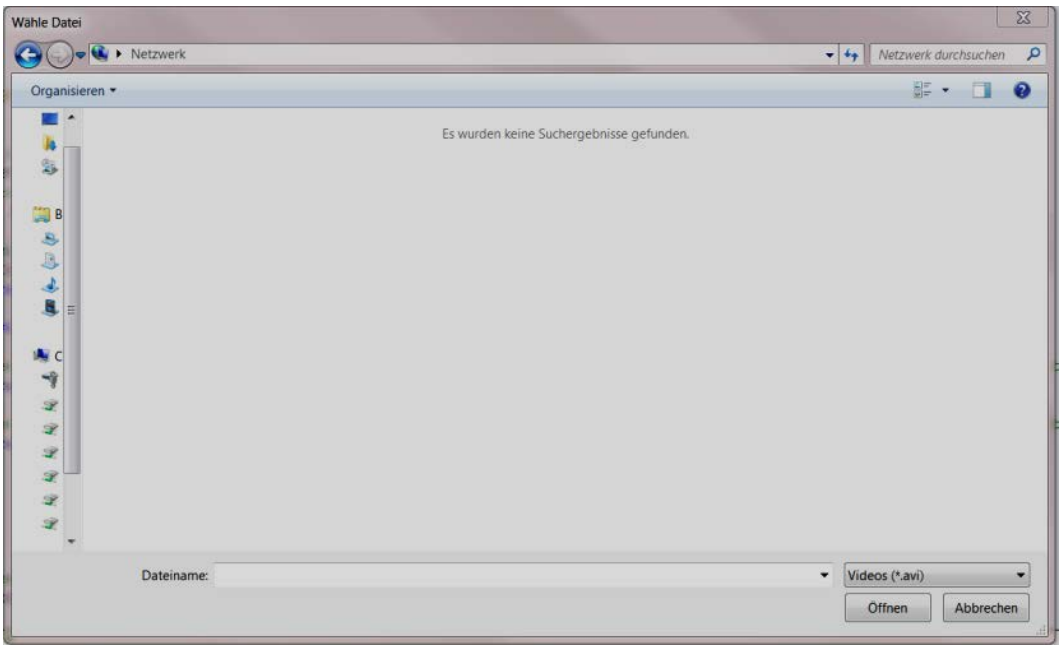
**FileOpenDialog**
Öffnet den Datei öffnen Dialog, um eine Datei auszuwählen.

Aufbau:

```
FileOpenDialog (“Title“, “init dir“, “filter“)
```

Beispiel:

```
Func _openVideoFile()

  FileOpenDialog.
    Local Const $sMessage = "Wähle Datei"

     $sFileOpenDialog = FileOpenDialog($sMessage, @WindowsDir & "\",
     "Videos (*.avi)", $FD_FILEMUSTEXIST + $FD_MULTISELECT)
  If @error Then

    MsgBox(4144,"Fehlermeldung", "Keine Datei gewählt!")

      FileChangeDir(@ScriptDir)
  Else

      FileChangeDir(@ScriptDir)

  EndIf

EndFunc
```



**FileSelectFolder**
FileSelectFolder öffnet die Ordnerauswahl.

Aufbau:

```
FileSelectFolder (“dialog text“, “root dir“)
```

Beispiel:

```
Local $var = FileSelectFolder("Ordner wählen", "")
```

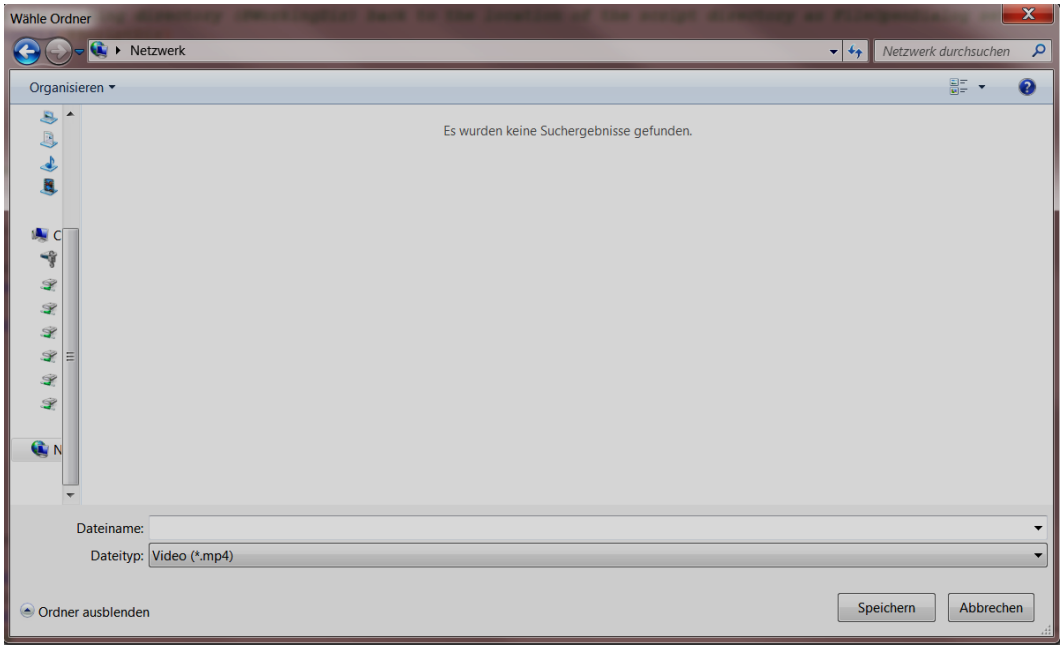
**FileSaveDialog**
Zeigt den Datei speichern Dialog.

Aufbau:

```
FileSaveDialog (“title“, „init dir“, “filter“)
```

Aufbau:

```
Func _FileSelectFolder()

 Global $MyDocsFolder = "::{450D8FBA-AD25-11D0-98A8-0800361B1103}"

  $sFileSelectFolder = FileSaveDialog("Wähle Ordner",  $MyDocsFolder, "
  "Video (*.mp4)", 2)
 If @error Then

   MsgBox(4144,"Fehlermeldung", "Kein Ziel gewählt!")

 EndIf
EndFunc
```



**FileWrite**
Schreibt einen Text in eine Textdatei.

Aufbau:

```
FileWrite(“filehandle“, “text“)
```

Beispiel:

```
Local $file = FileOpen("test.txt")

FileWrite($file, "Zeile 1")
FileWrite($file, "Immer noch Zeile 1" & @CRLF)
FileWrite($file, "Zeile 2")
```

**FileClose**
Schließt eine zuvor geöffnete Textdatei.

Aufbau:

```
FileClose (“filename“)
```

Beispiel:

```
Local $file = FileOpen("test.txt")

FileWrite($file, "Zeile 1")
FileWrite($file, "Immer noch Zeile 1" & @CRLF)
FileWrite($file, "Zeile 2")

FileClose($file)
```

**FileCopy**
Kopiert verschiedene Dateien.

Aufbau:

```
FileCopy (“source“, “dest“)
```

Beispiel:

```
FileCopy (“C:\test.txt“, “D:\testfolder\”)
```

**FileDelete**
Löscht verschiedene Dateien.

Aufbau:

```
FileDelete (“path“)
```

Beispiel:

```
FileDelete (“D:\testfolder\test.txt”)
```

**FileExists**
Prüft, ob eine Datei oder ein Verzeichnis besteht.

Aufbau:

```
FileExists (“path“)
```

Beispiel:

```
FileExists (“D:\testfolder\“)
FileExists (“C:\test.txt“)
```

**FileRecycle**
Löscht verschiedene Dateien und Ordner.

Aufbau:

```
FileRecycle (“source“)
```

Beispiel:

```
FileRecycle (“D:\testfolder\“)
FileRecycle (“C:\test.txt“)
```

## Versionsnummer

Versionsnummern unterscheiden einzelne Versionen einer Software, um deren Weiterentwicklungen nachvollziehbar zu kennzeichnen.

Die Versionsnummer ist die Grundlage für die Versionsverwaltung. Den Prozess der Vergabe der Versionsnummer nennt man Versionierung.

**Hauptversionsnummer:**

Indiziert meist äußerst signifikante Änderung am Programm – zum Beispiel wenn das Programm komplett neu geschrieben wurde (zum Beispiel GIMP 2.x nach der Version 1.x).

**Nebenversionsnummer:**

Bezeichnet meistens die funktionale Erweiterung des Programms.

**Revisionsnummer:**

Enthält meist Fehlerbehebungen.

**Buildnummer:**

Kennzeichnet in der Regel den Fortschritt der Entwicklungsarbeit in Einzelschritten, wird also zum Beispiel bei 0001 beginnend mit jedem Kompilieren des Codes um eins erhöht. Version 5.0.0.0042 stünde also für das 42. Kompilationsprodukt einer Software.

Beispiel:

**2.3.5.0041**

| | | └──── **Buildnummer**

| | └──── **Revisionsnummer**

| └──── **Nebenversionsnummer**

└──── **Hauptversionsnummer**