Dokumentation

zum

WizHook-Objekt

in

Microsoft Access 2000, 2002, 2003, 2007 u. 2010

von

Thomas Möller
Access@Team-Moeller.de

Stand: 12.12.2011

Inhaltsverzeichnis

1 Was ist das WizHook-Objekt? 4
2 Wie findet man das WizHook-Objekt? 4
3 Was bedeutet undokumentiert? 4
4 Das wichtigste: Der Key 5
5 Übersicht der Funktionen 5
5.1 AccessUserDataDir 6
5.2 AccessWizFilePath 7
5.3 AdpUIDPwd 8
5.4 AnalyzeQuery 9
5.5 AnalyzeTable 10
5.6 ArgsOfActid 12
5.7 BracketString 13
5.8 CacheStatus 14
5.9 CloseCurrentDatabase 15
5.10 CreateDataPageControl 16
5.11 CurrentLangID 17
5.12 DbcVbProject 18
5.13 EmbedFileOnDataPage 19
5.14 EnglishPictToLocal 20
5.15 ExecuteTempImexSpec 21
5.16 FCacheStatus 22
5.17 FCreateNameMap 23
5.18 FGetMSDE 24
5.19 FileExists 25
5.20 FirstDbcDataObject 26
5.21 FIsFEWch 27
5.22 FIsPublishedXasTable 28
5.23 FIsValidXasObjectName 29
5.24 FIsXasDb 30
5.25 FullPath 31
5.26 GetAccWizRCPath 32
5.27 GetAdeRegistryPath 33
5.28 GetColumns 34
5.29 GetCurrentView 35
5.30 GetDisabledExtensions 36
5.31 GetFileName 37
5.32 GetFileName2 39
5.33 GetFileOdso 40
5.34 GetImexTblName 41
5.35 GetInfoForColumns 42
5.36 GetLinkedListProperty 43
5.37 GetObjPubOption 44
5.38 GetScriptString 45
5.39 GetWizGlob 47
5.40 GlobalProcExists 48
5.41 HideDates 49
5.42 IsMatchToDbcConnectString 50
5.43 IsMemberSafe 51
5.44 IsValidIdent 52
5.45 Key 53
5.46 KeyboardLangID 54

5.47 KnownWizLeaks 55
5.48 LoadImexSpecSolution 56
5.49 LocalFont 57
5.50 NameFromActid 58
5.51 ObjTypOfRecordSource 59
5.52 OfficeAddInDir 60
5.53 OpenEmScript 61
5.54 OpenPictureFile 62
5.55 OpenScript 63
5.56 ReportLeaksToFile 65
5.57 SaveObject 66
5.58 SaveScriptString 67
5.59 SetDefaultSpecName 69
5.60 SetDpBlockKeyInput 70
5.61 SetVbaPassword 71
5.62 SetWizGlob 72
5.63 SortStringArray 73
5.64 SplitPath 74
5.65 TableFieldHasUniqueIndex 75
5.66 TranslateExpression 76
5.67 TwipsFromFont 77
5.68 WizCopyCmdbars 79
5.69 WizHelp 80
5.70 WizMsgBox 81
6 Anregungen und Ergänzungen 82
7 Links und andere Quellen 82
8 Anhang 83
8.1 Action-ID's und ihre Aktionen 83

1 Was ist das WizHook-Objekt?

Bereits seit der ersten Version ist Microsoft Access ist sehr erfolgreich im Markt für Desktop-Datenbanken. Ein wesentlicher Faktor dieses Erfolges ist es, dass auch unerfahrene Anwender sehr schnell ein ansprechendes Ergebnis erzielen. Dabei wird der Anwender von Assistenten – so genannten Wizards – unterstützt.

Ein Teil der Funktionalität der Oberfläche von Access findet sich in den Dateien „utility.mda", „acwzdat.mdt", „acwzusr.mdt", „acwzlib.mde", „acwzmain.mde" und „acwztool.mde". Diese hat Microsoft zum Teil freigegeben so dass jeder, der sich dafür interessiert, anschauen kann, wie die Assistenten funktionieren.

Ein anderer Teil der Funktionalität der Oberfläche von Access findet sich im WizHook-Objekt. Dieses Objekt stellt einiges an Basisfunktionalität bereit und ist nicht dokumentiert (siehe unten).

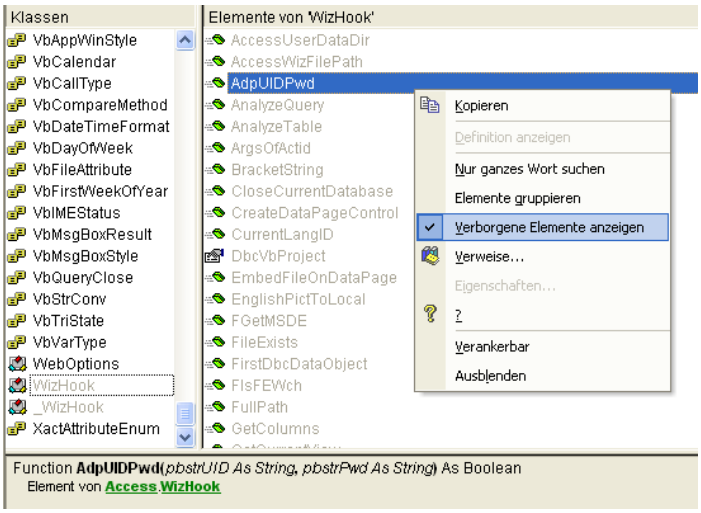
2 Wie findet man das WizHook-Objekt?

Um an das WizHook-Objekt mit seinen Eigenschaften und Methoden heranzukommen muss man in den VBA-Editor wechseln.

Als nächstes muss der Objektkatalog aktiviert werden. Dazu verwendet man den Befehl „Ansicht" / „Objektkatalog". Als Shortcut kann man auch die F2-Taste verwenden.

Im Objektkatalog werden alle Objekte mit ihren Eigenschaften und Methoden dargestellt. Ein Objekt mit dem Namen „WizHook" ist aber nicht zu finden.

Um das WizHook-Objekt sichtbar zu machen muss man die verborgenen Elemente anzeigen. Dazu klickt man mit der rechten Maustaste irgendwo, wo man keine Auswahl trifft. Aus dem Kontextmenü wählt man „Verborgene Elemente anzeigen". Jetzt werden die versteckten Objekte eingeblendet. Ganz am Ende der Liste findet sich auch das WizHook-Objekt.



3 Was bedeutet undokumentiert?

Beim Einsatz des WizHook-Objekts und seiner Funktionen ist zu beachten, dass das WizHook-Objekt nicht offiziell dokumentiert ist. Das bedeutet, dass es keine Unterstützung von Seiten von Microsoft gibt. Sie finden keine offizielle Dokumentation. Auch eine Hilfe zum WizHook-Objekt gibt es nicht. (Deshalb gibt es ja dieses Dokument ☺.)

Aus dem Umstand, das Microsoft das WizHook-Objekt nicht dokumentiert hat, ergibt sich das Risiko, dass in einer zukünftigen Version die Eigenschaften und Methoden des Objekts geändert werden. Es kann auch passieren, dass eine Eigenschaft oder Methode nicht mehr implementiert ist. Der von Ihnen erstellte Code ist dann u. U. nicht mehr lauffähig oder muss ggfs. aufwändig angepasst werden.

Bei der Verwendung des WizHook-Objekts ist Vorsicht geboten. Wenn Sie einen Parameter falsch übergeben kann es sein, dass Access komplett abstürzt. Auch dies ergibt sich aus der Tatsache, dass das WizHook-Objekt nicht dokumentiert ist. Die Funktionen sind nicht für die Allgemeinheit

implementiert worden. Vielmehr wird davon ausgegangen, dass der Programmierer weiß, was er tut.

Trotz all dieser Risiken erscheinen manche Funktionen sehr reizvoll. Man muss sich dieser Risiken bei der Verwendung des WizHook-Objekts bewusst sein. Auf jeden Fall sollte man durch geeigneten Code sicherstellen, dass die übergebenen Parameter in den jeweiligen Gültigkeitsbereich passen.

Abschließend möchte ich nochmals betonen: Der Einsatz erfolgt auf eigenes Risiko.

4 Das wichtigste: Der Key

Es haben schon viele Programmierer das WizHook-Objekt mit seinen Eigenschaften und Methoden im Objekt-Katalog entdeckt. Beim Einsatz wurden viele jedoch enttäuscht. Die Funktionen des WizHook-Objekts schienen nicht zu funktionieren. Es passierte einfach nicht.

Der Schlüssel zum WizHook-Objekt ist der Key. Damit wird das WizHook-Objekt aktiviert. Der Key muss während einer Access-Sitzung nur einmal eingegeben werden und schon stellt das WizHook-Objekt seine Eigenschaften und Methoden während der gesamten Laufzeit von Access zur Verfügung.

Der Wert, der der Eigenschaft „Key" übergeben werden muss, lautet: **51488399**.

Es gibt verschiedene Strategien, um den Key zu setzen:

- Eine Möglichkeit besteht darin, den Key direkt nach dem Start von Access zu setzen. Der Vorteil liegt darin, dass man sich bei der Verwendung der Methoden des WizHook-Objekts nicht mehr um den Key kümmern muss.
- Eine weitere Möglichkeit besteht darin, den Key direkt vor der Verwendung des WizHook-Objekts zu setzen und ihn direkt nach dessen Verwendung wieder zurück zu setzen.
- Die dritte Möglichkeit besteht darin, den Key immer direkt vor der Verwendung des WizHook-Objekts zu setzen und auf das Zurücksetzen zu verzichten.

Ich persönlich bevorzuge den dritten Weg. Der Code wird dadurch klarer und verständlicher. Der Key wird an der Stelle gesetzt, an der er benötigt wird. Auf das (überflüssige) Zurücksetzen des Keys wird verzichtet.

5 Übersicht der Funktionen

Es folgt eine Auflistung aller Funktionen, die in Access 2000, 2002, 2003 und 2007 verfügbar sind.

Zu jeder Funktion wird genannt, ab welcher Access-Version sie einsetzbar ist. Der Zweck der Funktion wird kurz beschrieben.
Die Deklaration, die Argumente und die Rückgabewerte beschreiben den syntaktischen Aufbau der Funktion.
Ein Codebeispiel soll den Einsatz verdeutlichen. Ggfs. runden weitere Hinweise die Beschreibung ab.

5.1 AccessUserDataDir

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Liefert den Pfad zum Access-Datenverzeichnis.

Deklaration
Function AccessUserDataDir() As String

Argumente
- - -

Rückgabewert
Pfad zum Access-Datenverzeichnis.
Bei mir z.B.: C:\Dokumente und Einstellungen\Thomas\Anwendungsdaten\Microsoft\Access\

Code-Beispiel
```
WizHook.Key = 51488399
MsgBox WizHook.AccessUserDataDir
```

Hinweise
- - -

5.2 AccessWizFilePath

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
Liefert den vollständigen Pfad zu einer Wizard-Datenbank.

Deklaration
Function AccessWizFilePath(bstrWhich As String) As String

Argumente
bstrWhich Name der Wizard-Datenbank

Rückgabewert
Vollständiger Pfad zur Wizard-Datenbank.
Bei mir z.B.: C:\Programme\ Microsoft Office XP\Office10\utility.mda

Code-Beispiel
```
Dim strWizard As String

strWizard = "utility.mda"

WizHook.Key = 51488399
MsgBox WizHook.AccessWizFilePath(strWizard)
```

Hinweise
Wenn die Wizard-Datenbank nicht existiert, liefert die Funktion einen leeren String zurück.
Ich habe die Funktion für folgende Wizard-Datenbanken getestet:
utility.mda, acwzdat.mdt, acwzusr.mdt, acwzlib.mde, acwzmain.mde und acwztool.mde.

5.3 AdpUIDPwd

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck

Ermittelt, ob es sich um eine ADP-Datei handelt und gibt die User-ID und das Passwort des aktiven Users (Access-Sicherheitssystem) aus.

Deklaration

Function AdpUIDPwd(pbstrUID As String,
pbstrPwd As String) As Boolean

Argumente

pbstrUID	User-ID des aktiven Users
pbstrPwd	Passwort des aktiven Users

Rückgabewert

True: Bei der Datenbank handelt es sich um ein ADP.
False: Bei dieser Datenbank handelt es sich nicht um eine ADP.

Code-Beispiel

```
Dim fErgebnis As Boolean
Dim strUserID As String
Dim strPasswort As String

WizHook.Key = 51488399

fErgebnis = WizHook.AdpUIDPwd(strUserID, strPasswort)

MsgBox "Ist ADP: " & fErgebnis & vbCrLf & _
       vbCrLf & _
       "User-ID: " & strUserID & vbCrLf & _
       vbCrLf & _
       "Passwort: " & strPasswort
```

Hinweise

- - -

Ich danke Serge Gavrilov für den entscheidenden Hinweis.

5.4 AnalyzeQuery

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Mit dieser Funktion untersucht der „Assistent zur Leistungsanalyse" Abfragen.

Deklaration

```
Function AnalyzeQuery(Workspace As Workspace,
                      Database As Database,
                      Query As String,
                      Results As String) As Long
```

Argumente

Workspace	DAO.Workspace-Objekt, in dem sich die Datenbank befindet
Database	DAO.Datenbank-Objekt, das die zu untersuchende Abfrage enthält
Query	Name der Abfrage
Results	Hier werden die Ergebnisse der Analyse zurückgegeben. Die einzelnen Teile des Strings sind durch Chr$(1) voneinander getrennt.

Rückgabewert

Scheinbar ist der Rückgabewert immer „0".

Code-Beispiel

```
Dim pdbe As DAO.PrivDBEngine
Dim db As DAO.Database
Dim wrk As DAO.Workspace
Dim strQuery As String
Dim strResults As String
Dim lngErgebnis As Long
Dim astrErgebnis() As String
Dim intI As Integer

'Initialisieren
Set pdbe = New DAO.PrivDBEngine
Set wrk = pdbe.CreateWorkspace("MyWorkSpace", "Admin", "", dbUseJet)
Set db = wrk.OpenDatabase(Me!txtPfad & Me!txtNameDB)
strQuery = Me!txtAbfrage

'Ergebnis ermitteln
WizHook.Key = 51488399
lngErgebnis = WizHook.AnalyzeQuery(wrk, db, strQuery, strResults)

'Ausgabe der Ergebnisse
If Len(strResults) > 0 Then
    astrErgebnis() = Split(strResults, Chr$(1))
    For intI = 0 To UBound(astrErgebnis())
        MsgBox astrErgebnis(intI), vbInformation, "Ergebnis: " & lngErgebnis
    Next intI
Else
    MsgBox "Der Analyzer liefert keine Ergebnisse", vbExclamation
End If

db.Close
Set db = Nothing
```

```
wrk.Close
Set wrk= Nothing
Set pdbe = Nothing
```

Hinweise

- - -

5.5 AnalyzeTable

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Mit dieser Funktion untersucht der „Assistent zur Leistungsanalyse" Tabellen.

Deklaration

```
Function AnalyzeTable(Workspace As Workspace,
                      Database As Database,
                      Table As String,
                      ReturnDebugInfo As Boolean,
                      Results As String) As Long
```

Argumente

Workspace	DAO.Workspace-Objekt, in dem sich die Datenbank befindet
Database	DAO.Datenbank-Objekt, das die zu untersuchende Tabelle enthält
Table	Name der Tabelle
ReturnDebugInfo	???
Results	Hier werden die Ergebnisse der Analyse zurückgegeben. Die einzelnen Teile des Strings sind durch Chr$(1) voneinander getrennt.

Rückgabewert

Scheinbar ist der Rückgabewert immer „0".

Code-Beispiel

```
Dim pdbe As DAO.PrivDBEngine
Dim db As DAO.Database
Dim wrk As DAO.Workspace
Dim strTable As String
Dim fInfo As Boolean
Dim strResults As String
Dim astrErgebnis() As String
Dim lngErgebnis As Long
Dim intI As Integer

'Initialisieren
Set pdbe = New DAO.PrivDBEngine
Set wrk = pdbe.CreateWorkspace("MyWorkSpace", "Admin", "", dbUseJet)
Set db = wrk.OpenDatabase(Me!txtPfad & Me!txtNameDB)
strTable = "Me!txtTabelle

'Ergebnis ermitteln
WizHook.Key = 51488399
lngErgebnis = WizHook.AnalyzeTable(wrk, db, strTable, fInfo, strResults)
```

```
'Ergebnisse anzeigen
If Len(strResults) > 0 Then
    astrErgebnis() = Split(strResults, Chr$(1))
    For intI = 0 To UBound(astrErgebnis())
        MsgBox astrErgebnis(intI), vbInformation, "Ergebnis: " & lngErgebnis & _
               " / " & fInfo
    Next intI
Else
    MsgBox "Der Analyzer liefert keine Ergebnisse", vbExclamation
End If

db.Close
Set db = Nothing
wrk.Close
Set wrk= Nothing
Set pdbe = Nothing
```

Hinweise

- - -

5.6 ArgsOfActid

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Liefert die Zahl der Argumente für eine Action-ID

Deklaration
Function ArgsOfActid(Actid As Long) As Long

Argumente
Actid Zahlenwert, der die Action-ID angibt

Rückgabewert
Anzahl der erwarteten Argumente

Code-Beispiel
```
Dim lngActID As Long

lngActID = CLng(Nz(Me!txtActID, 0))

WizHook.Key = 51488399
MsgBox "Action-ID: " & lngActID & vbCrLf & _
       "Name : " & WizHook.NameFromActid(lngActID) & vbCrLf & _
       "Argumente: " & WizHook.ArgsOfActid(lngActID)
```

Hinweise
Siehe auch „NameFromActid" und „Action-ID's und ihre Aktionen".

Als Action-ID können Werte zwischen 1 und 65 eingegeben werden. Alle anderen Werte liefern „0" zurück.

5.7 BracketString

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Setzt die einzelnen Teile eines Ausdruck in eckige Klammern („[]")

Deklaration

Function BracketString(String As String,
flags As Long) As Boolean

Argumente

String	zu ergänzender Ausdruck
flags	0 : Es werden keine Klammern hinzugefügt. 1 : Klammern werden nur hinzugefügt, wenn der Ausdruck einen ungültigen Variablennamen enthält. 2 : Es werden immer Klammern hinzugefügt.

Rückgabewert

True, wenn der Ergebnisstring einen **un**gültigen Ausdruck ergibt.
False, wenn der Ergebnisstring einen gültigen Ausdruck ergibt.

Code-Beispiel

```
Dim strAusdruck As String

strAusdruck = Nz(Me!txtAusdruck, "")

WizHook.Key = 51488399
If WizHook.BracketString(strAusdruck, Me!fraFlags) = True Then
    MsgBox "Ergebnis: Ungültiger Ausdruck!" & vbCrLf & vbCrLf & strAusdruck, _
           vbInformation, "True"
Else
    MsgBox "Ergebnis: Gültiger Ausdruck!" & vbCrLf & vbCrLf & strAusdruck, _
           vbInformation, "False"
End If
```

Hinweise

Es wird nicht geprüft, ob das Objekt, das im Ausdruck angesprochen wird, existiert.

5.8 CacheStatus

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck
???

Deklaration
Sub CacheStatus(bstrStatus As String)

Argumente
bstrStatus ???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.9 CloseCurrentDatabase

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Die aktuelle Datenbank wird geschlossen.

Deklaration
Function CloseCurrentDatabase() As Boolean

Argumente
- -

Rückgabewert
True, wenn die Aktion erfolgreich war.
False, wenn die Aktion fehlgeschlagen ist.

Code-Beispiel
```
WizHook.Key = 51488399
WizHook.CloseCurrentDatabase
```

Hinweise
Bei meinen Test wurde die aktuelle Datenbank nicht geschlossen.
Statt dessen wurde lediglich das aktuelle Datenbankobjekt geschlossen.

Mit dem Befehl >>Application.CloseCurrentDatabase<< wird die aktuelle Datenbank wie gewünscht geschlossen.

5.10 CreateDataPageControl

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Erstellt ein neues Steuerelement auf einer Datenzugriffsseite

Deklaration

```
Sub CreateDataPageControl(DpName As String,
                          CtlName As String,
                          Typ As Long,
                          Section As String,
                          SectionType As Long,
                          AppletCode As String,
                          X As Long,
                          Y As Long,
                          dx As Long,
                          dy As Long)
```

Argumente

DpName	Name der Datenzugriffsseite
CtlName	Name des (neuen) Steuerelements
Typ	Steuerelementtyp (wie in der Enumeration AcControlType aufgezählt)
Section	Section, in der das Steuerelement angelegt werden soll (wie in der Enumeration AcSection aufgezählt.)
SectionType	???
AppletCode	???
X	Abstand vom linken Rand
Y	Abstand vom oberen Rand
dx	Breite
dy	Höhe

Rückgabewert

- - -

Code-Beispiel

```
DoCmd.OpenDataAccessPage "Seite1", acDataAccessPageDesign

WizHook.Key = 51488399
WizHook.CreateDataPageControl "Seite1", "TEST", acTextBox, acDetail, -1, "",_
                              500, 500, 2000, 200
```

Hinweise

Die Datenzugriffsseite muss vorher in der Entwurfsansicht geöffnet sein.
Für den Parameter „SectionType" führen Werte größer als 4 zu einer Fehlermeldung. Bei meinen Test führte nur der Wert „-1" zum Erfolg.

5.11 CurrentLangID

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Ermittelt die ID der aktuell verwendeten Sprache.

Deklaration
Function CurrentLangID() As Long

Argumente
- - -

Rückgabewert
ID der aktuell verwendeten Sprache, z.B. 1031 für Deutsch.

Code-Beispiel
```
WizHook.Key = 51488399
MsgBox WizHook.CurrentLangID
```

Hinweise
- - -

5.12 DbcVbProject

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Liefert einen Verweis auf das aktuelle VB-Projekt.

Deklaration
Property DbcVbProject As VBProject

Argumente
- - -

Rückgabewert
Verweis auf das aktuelle VB-Projekt.

Code-Beispiel
```
Dim obj As Object

WizHook.Key = 51488399
Set obj = WizHook.DbcVbProject

MsgBox obj.Name

Set obj = Nothing
```

Hinweise
Es wird kein Verweis auf die „Microsoft Visual Basic for Applications Extensibility"-Library benötigt.

5.13 EmbedFileOnDataPage

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
???

Deklaration
Function EmbedFileOnDataPage(DpName As String,
FileToInsert As String) As String

Argumente

DpName	???
FileToInsert	???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.14 EnglishPictToLocal

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Übersetzt die Ausdrücke aus der Format-Eigenschaft in die aktuelle Sprache

Deklaration

Function EnglishPictToLocal(In As String,
Out As String) As Boolean

Argumente

In	Eingabewert: zu übersetzender Ausdruck
Out	Rückgabewert: Übersetzung des Ausdrucks in aktueller Sprache

Rückgabewert

Der Rückgabewert scheint immer „False" zu sein.

Code-Beispiel

```
Dim ctl As Control
Dim strIn As String
Dim strOut As String
Dim fSuccess As Boolean

For Each ctl In Me!fraAusdruck.Controls
    If ctl.ControlType = acOptionButton Then
        If ctl.OptionValue = Me!fraAusdruck Then
            strIn = ctl.Controls(0).Caption
            Exit For
        End If
    End If
Next

WizHook.Key = 51488399
fSuccess = WizHook.EnglishPictToLocal(strIn, strOut)

MsgBox "Eingabewert: " & strIn & vbCrLf & vbCrLf & _
       "Ausgabewert: " & strOut, vbInformation, "Erfolgreich: " & fSuccess
```

Hinweise

Wenn als Eingabewert ein Text verwendet wird, der nicht als Ausdruck in der Format-Eigenschaft zur Verfügung steht, versucht die Funktion trotzdem eine (eher sinnlose) Übersetzung zu liefern.

Es ist mir in meinen Tests (mit Access 2002) (bisher) nicht gelungen, als Rückgabewert „True" zu erzeugen.

5.15 ExecuteTempImexSpec

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck
???

Deklaration
Sub ExecuteTempImexSpec(bstrSpecXML As String)

Argumente
bstrSpecXML ???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.16 FCacheStatus

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck
???

Deklaration
Function FCacheStatus() As Boolean

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.17 FCreateNameMap

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2003, 2007, 2010

Funktion/Zweck
???

Deklaration
```
Function FCreateNameMap(objtyp As Integer,
                        bstrObjName As String) As Boolean
```

Argumente

objtyp	???
bstrObjName	???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.18 FGetMSDE

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Function FGetMSDE(fBlockKeys As Boolean) As Boolean

Argumente
fBlockKeys ???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.19 FileExists

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Prüft, ob eine Datei existiert.

Deklaration
Function FileExists(File As String) As Boolean

Argumente

File zu prüfender Dateiname incl. Pfadangabe

Rückgabewert
True, wenn die Datei existiert
False, wenn die Datei nicht existiert.

Code-Beispiel

```
Dim strDateiName As String

strDateiName = Nz(Me!txtDateiName, "")

WizHook.Key = 51488399
MsgBox WizHook.FileExists(strDateiName)
```

Hinweise
Die Funktion prüft leider nicht, ob ein Ordner existiert. Wenn der Name eines Ordners übergeben wird, liefert die Funktion immer „False" zurück.

5.20 FirstDbcDataObject

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Liefert Informationen über das erste Datenobjekt der Datenbank.

Deklaration

Function FirstDbcDataObject(Name As String,
ObjType As AcObjectType,
Attribs As Long) As Boolean

Argumente

Name	Name des Objekts
ObjType	Typ des Objekts: 0 für Tabelle 1 für Abfrage
Attribs	Attribute des Objekts: 0: Normal 8: Versteckt 2097152: Verknüpft

Rückgabewert

True, wenn die Datei existiert
False, wenn die Datei nicht existiert.

Code-Beispiel

```
Dim strName As String
Dim lngTyp As Long
Dim lngAttribut As Long

WizHook.Key = 51488399
If WizHook.FirstDbcDataObject(strName, lngTyp, lngAttribut) = True Then
    MsgBox "Name: " & strName & vbCrLf & _
           "Typ: " & lngTyp & vbCrLf & _
           "Attribute: " & lngAttribut, vbInformation
Else
    MsgBox "Aktion fehlgeschlagen", vbCritical
End If
```

Hinweise

Die Ergebnisse der Funktion werden über die zu übergebenden Argumente zurückgegeben.

5.21 FIsFEWch

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Function FIsFEWch(wch As Long) As Boolean

Argumente
wch ???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.22 FIsPublishedXasTable

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2010

Funktion/Zweck
???

Deklaration
Function FIsPublishedXasTable(bstrObjectName As String) As Boolean

Argumente
bstrObjectName ???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.23 FIsValidXasObjectName

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2010

Funktion/Zweck
???

Deklaration
Function FIsValidXasObjectName(bstrObjectName As String,
iobjtyp As AcObjectType) As Boolean

Argumente

bstrObjectName	???
iobjtyp	???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.24 FIsXasDb

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2010

Funktion/Zweck
???

Deklaration
Function FIsXasDb() As Boolean

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.25 FullPath

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Wandelt eine relative Pfadangabe in einen absoluten Pfad um.

Declaration
Function FullPath(RelativePath As String,
FullPath As String) As Integer

Argumente
RelativePath	Relativer Pfad
FullPath	Hier wird der absolute Pfad zurückgegeben.

Rückgabewert
True, wenn ein absoluter Pfad zurückgegeben werden kann.
False, wenn kein absoluter Pfad zurückgegeben werden kann, z.B. wenn als relativer Pfad nichts übergeben wird.

Code-Beispiel
```
Dim strRelPfad As String
Dim strAbsPfad As String

strRelPfad = Nz(Me!txtRelPfad, "")

WizHook.Key = 51488399
If WizHook.FullPath(strRelPfad, strAbsPfad) = True Then
    MsgBox strAbsPfad, vbInformation
Else
    MsgBox "Aktion fehlgeschlagen", vbCritical
End If
```

Hinweise
Ausgangspunkt für die Umwandlung ist immer das „CurDir".

Es sind die (aus HTML) bekannten Kennzeichen für übergeordnete Ordner möglich („../").

Bei der Ermittlung des absoluten Pfad wird nicht geprüft, ob der Ordern wirklich vorhanden ist.

5.26 GetAccWizRCPath

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Function GetAccWizRCPath() As String

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.27 GetAdeRegistryPath

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck
????

Deklaration
Function GetAdeRegistryPath() As String

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.28 GetColumns

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
Die Funktion liefert eine durch Semikolon getrennte Liste der Felder ein Tabelle oder Abfrage.

Deklaration
Function GetColumns(bstrBase As String) As String

Argumente
bstrBase Name einer Tabelle, einer Abfrage oder ein SQL-String

Rückgabewert
Eine durch Semikolon getrennte Liste der Felder.

Code-Beispiel
```
Dim strTabelle As String

strTabelle = Nz(Me!txtTabelle, "")

WizHook.Key = 51488399
MsgBox WizHook.GetColumns(strTabelle)
```

Hinweise
Wenn die Tabelle oder Abfrage nicht existiert liefert die Funktion einen leeren String zurück.

5.29 GetCurrentView

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
Ermittelt, in welcher Ansicht eine Tabelle angezeigt wird.

Deklaration
Function GetCurrentView(bstrTableName As String) As Long

Argumente
bstrTableName Name der Tabelle

Rückgabewerte
Es ist anzunehmen, dass Werte, wie sie in der Enumeration „AcCurrentView" aufgezählt sind, zurückgegeben werden. (Näheres siehe Hinweise.)

Code-Beispiel
```
Dim strTabelle As String

strTabelle = Nz(Me!txtTabelle, "")

WizHook.Key = 51488399
MsgBox WizHook.GetCurrentView(strTabelle)
```

Hinweise
Diese Funktion scheint nicht zu funktionieren.

Die Funktion liefert immer den Wert „2" zurück. Das Ergebnis ist dabei unabhängig von der Ansicht der Tabelle.
Wenn die übergebene Tabelle nicht existiert liefert die Funktion „0" zurück.

5.30 GetDisabledExtensions

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2010

Funktion/Zweck
???

Deklaration
Function GetDisabledExtensions() As String

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.31 GetFileName

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Zeigt den Office-Dateiauswahl-Dialog an (ohne das ein Verweis auf die Office-Bibliothek erforderlich ist).

Deklaration

```
Function GetFileName(hWndOwner As Long,
                     AppName As String,
                     DlgTitle As String,
                     OpenTitle As String,
                     File As String,
                     InitialDir As String,
                     Filter As String,
                     FilterIndex As Long,
                     View As Long,
                     flags As Long,
                     fOpen As Boolean) As Long
```

Argumente

hWndOwner	Zeiger auf das Fenster, das den Dialog gestartet hat
AppName	???
DlgTitle	Dialog-Titel, Beschriftung der Titelzeile
OpenTitle	Beschriftung des Öffnen-Buttons
File	Gibt Pfad und Namen der ausgewählten Datei(en) oder Ordner zurück
InitialDir	Startverzeichnis
Filter	Liste mit Filtern für den Dateityp. Einzelne Einträge sind durch das Pipe-Zeichen („|") von einander getrennt.
FilterIndex	Nummer des Filters, der bei der Anzeige aktiv ist (0-basiert)
View	0: Detailansicht 1: Vorschauansicht 2: Eigenschaften 3: Liste 4: Miniaturansicht 5: Große Symbole 6: Kleine Symbole
flags	Als Auswahl für diesen Parameter können folgende Werte addiert werden: 4: Set Current Dir 8: Mehrfachauswahl möglich 32: Ordnerauswahldialog 64: Wert im Parameter „View" berücksichtigen
fOpen	True: Öffnenschaltfläche öffnet ausgewählten Ordner False: Öffnenschaltfläche öffnet ausgewählten Ordner nicht

Rückgabewerte

0, wenn eine Datei ausgewählt wurde.
-302, wenn der Dialog (ohne Auswahl) abgebrochen wurde.

Code-Beispiel

```
Dim hwndOwner As Long
Dim strAppName As String
Dim strDlgTitle As String
Dim strOpenTitle As String
Dim strFile As String
Dim strInitialDir As String
Dim strFilter As String
Dim lngFilterIndex As Long
Dim lngView As Long
Dim lngflags As Long
Dim fOpen As Boolean
Dim intI As Integer

'Werte übernehmen
hwndOwner = Me.Hwnd
strDlgTitle = Nz(Me!txtDialogTitel, "")
strOpenTitle = Nz(Me!txtOpenTitel, "")
strFile = ""
strInitialDir = Nz(Me!txtStartVerzeichnis, "")
strFilter = Nz(Me!txtFilter, "")
lngFilterIndex = Nz(Me!txtFilterIndex, 0)
lngView = Nz(Me!txtView, 0)
lngflags = Nz(Me!txtFlags, 0)
fOpen = Me!chkOpen

WizHook.Key = 51488399
If WizHook.GetFileName(hwndOwner, strAppName, strDlgTitle, strOpenTitle, _
        strFile, strInitialDir, strFilter, lngFilterIndex, lngView, _
        lngflags, fOpen) = 0 Then
    If InStr(strFile, vbTab) = 0 Then
        MsgBox strFile, vbInformation, "Auswahl:"
    Else
        intI = 1
        While InStr(intI + 1, strFile, vbTab) > 0
           MsgBox Mid$(strFile, intI + 1, InStr(intI + 1, strFile, vbTab) - intI)
           intI = InStr(intI + 1, strFile, vbTab)
        Wend
        MsgBox Right$(strFile, Len(strFile) - intI)
    End If
End If
```

Hinweise

Wenn kein Startverzeichnis angegeben wird, beginnt der Dialog im aktuellen Verzeichnis (CurDir). Wenn das Startverzeichnis nicht existiert, beginnt der Dialog im Ordner „Eigene Dateien".

Wenn kein Filter übergeben wird, verwendet der Dialog „Alle Dateien (*.*).
Wenn ein FilterIndex angegeben wird, der nicht vorhanden ist, wird der erste Filter verwendet.

Wenn ein Dialog zur Ordnerauswahl geöffnet wird, muss für den Parameter „fOpen" der Wert „True" übergeben werden. Sonst kann kein Ordner ausgewählt werden.

Die Auswahl für den Parameter „View" ist nur wirksam, wenn bei „Flags" der Wert 64 (View berücksichtigen) verwendet wird.

Das Kennzeichen „Mehrfachauswahl" wird nur berücksichtigt, wenn der Parameter „fOpen" den Wert „True" hat.

Mehrere Dateien werden durch „Tab" getrennt zurückgegeben.

5.32 GetFileName2

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck

Zeigt den Office-Dateiauswahl-Dialog an (ohne das ein Verweis auf die Office-Bibliothek erforderlich ist).

Deklaration

```
Function GetFileName2(hwndOwner As Long,
                      AppName As String,
                      DlgTitle As String,
                      OpenTitle As String,
                      File As String,
                      InitialDir As String,
                      Filter As String,
                      FilterIndex As Long,
                      View As Long,
                      flags As Long,
                      fOpen As Boolean,
                      fFileSystem) As Long
```

Argumente

Siehe „GetFileName"

Rückgabewerte

Siehe „GetFileName"

Code-Beispiel

Im Wesentlichen identisch mit „GetFileName".

Hinweise

Diese Funktion ist scheinbar identisch zur Funktion „GetFileName". Unterschiede konnte ich (noch) nicht herausfinden.

Der einzige sichtbar Unterschied liegt im zusätzlichen Parameter „fFileSystem". Die Auswirkungen bzw. der Sinn dieses Parameters haben sich mir bisher (noch) nicht erschlossen.

5.33 GetFileOdso

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck

Öffnen einen Auswahl-Dialog für eine Datenquelle und gibt den DSN-String zurück.

Deklaration

```
Function GetFileOdso(bstrExt As String,
                     bstrFilename As String) As Long
```

Argumente

bstrExt	???
bstrFilename	Hier wird der vollständige DSN-String zurückgegeben

Rückgabewerte

0, wenn Auswahl abgebrochen wurde.
2, wenn DSN-String zurückgegeben wurde.

Code-Beispiel

```
Dim strExt As String
Dim strFileName As String

WizHook.Key = 51488399
If WizHook.GetFileOdso(strExt, strFileName) <> 0 Then
    MsgBox strFileName
End If
```

Hinweise

Den Zweck des Parameters „bstrExt" konnte ich bisher nicht ermitteln.

5.34 GetImexTblName

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck
????

Deklaration
Function GetImexTblName() As String

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.35 GetInfoForColumns

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
Liefert eine durch Semikolon getrennte Liste der Felder und deren Attribute einer Tabelle oder Abfrage.

Deklaration
Function GetInfoForColumns(bstrBase As String) As String

Argumente
bstrBase Name einer Tabelle, einer Abfrage oder ein SQL-String

Rückgabewert
Eine durch Semikolon getrennte Liste der Felder und deren Attribute.
Für jedes Feld werden drei Werte ausgegeben:
- Feldname
- Felddatentyp als DAO-Datentypenkonstante (dbLong = 4, dbText = 10, etc.)
- Größe des Feldes

Code-Beispiel
```
Dim strTabelle As String

strTabelle = Nz(Me!txtTabelle, "")

WizHook.Key = 51488399
MsgBox WizHook.GetInfoForColumns(strTabelle)
```

Hinweise
Wenn die Tabelle oder Abfrage nicht existiert liefert die Funktion einen leeren String zurück.

5.36 GetLinkedListProperty

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck

????

Deklaration

```
Function GetLinkedListProperty(bstrTableName As String,
                               bstrPropertyName As String,
                               fServer As Boolean) As String
```

Argumente

bstrTableName	???
bstrPropertyName	???
fServer	???

Rückgabewert

???

Code-Beispiel

- - -

Hinweise

Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.37 GetObjPubOption

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2010

Funktion/Zweck
???

Deklaration
```
Function GetObjPubOption(bstrObjectName As String,
                         iobjtyp As AcObjectType,
                         fTablesAsClient As Boolean) As Long
```

Argumente

bstrObjectName	???
iobjtyp	???
fTablesAsClient	???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.38 GetScriptString

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2010

Funktion/Zweck

Liefert die Werte aus den verschiedenen Spalten eines Makros und seiner Argumente.

Deklaration

```
Function GetScriptString(HScr As Long,
                         ScriptColumn As Long,
                         Value As String) As Boolean
```

Argumente

HScr	Referenz auf ein geöffnetes Makro
ScriptColumn	Zahl, die die Spalte angibt 0: Name des Markos 1: Kommentar 2: Bedingung 3-12: Argumente
Value	Inhalt der Spalte (wird von Funktion zurückgeliefert)

Rückgabewert

True, wenn Funktion erfolgreich beendet wurde.
False, wenn Funktion fehlerhaft ausgeführt wurde.

Code-Beispiel

```
Dim hScr As Long
Dim strScript As String
Dim strLabel As String
Dim lngOpenMode As Long
Dim lngExtra As Long
Dim lngVersion As Long
Dim lngActionID As Long
Dim strAction As String
Dim strText As String
Dim fEnde As Boolean

'Initialisieren
strScript = Nz(Me!txtMakroName, "")
strLabel = "Test"
lngOpenMode = 0

'Makro öffnen
WizHook.Key = 51488399
hScr = WizHook.OpenScript(strScript, strLabel, lngOpenMode, lngExtra, lngVersion)

'Ausstieg bei Fehler
If hScr = 0 Then
    MsgBox "Markro konnte nicht geöffnet werden."
    Exit Sub
End If

'Makro lesen
Do While Not (fEnde)
```

```
    api_Makro_NextRow hScr, 0&, 0&
    lngActionID = api_Makro_GetActID(hScr)
    If lngActionID <> -1 Then
        strAction = WizHook.NameFromActid(lngActionID)
        MsgBox strAction, vbInformation, "Aktion:"
        WizHook.GetScriptString hScr, 0&, strText
        MsgBox strText, vbInformation, "Label:"
        WizHook.GetScriptString hScr, 1&, strText
        MsgBox strText, vbInformation, "Kommentar:"
        WizHook.GetScriptString hScr, 2&, strText
        MsgBox strText, vbInformation, "Bedingung:"
        WizHook.GetScriptString hScr, 3&, strText
        MsgBox strText, vbInformation, "Argument:"
    Else
        fEnde = True
    End If
Loop

'Makro schliessen
fCloseHscr hScr

'Zusätzlich müssen folgende Funktionen deklariert sein:

Private Declare Function api_Makro_NextRow _
        Lib "msaccess.exe" _
        Alias "#22" _
        (ByVal Makro As Long, _
        ByVal lngSkipBlank As Long, _
        lngEndOfMakro As Long) As Long
Private Declare Function api_Makro_GetActID _
        Lib "msaccess.exe" _
        Alias "#29" _
        (ByVal Makro As Long) As Long
Private Declare Sub fCloseHscr _
        Lib "msaccess.exe" _
        Alias "#20" _
        (ByVal hScr As Long)
```

Hinweise

Siehe auch „OpenScript", „SaveScriptString" und „NameFromActid".

5.39 GetWizGlob

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Function GetWizGlob(lWhich As Long)

Argumente
lWhich ???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.40 GlobalProcExists

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Prüft, ob eine als „public" deklarierte Prozedur in einem Standardmodul vorhanden ist.

Deklaration
Function GlobalProcExists(Name As String) As Boolean

Argumente
Name Name der Prozedur

Rückgabewert
True, wenn öffentliche Prozedur vorhanden ist.
False, wenn keine öffentliche Prozedur vorhanden ist.

Code-Beispiel
```
Dim strName As String

strName = Nz(Me!txtName, "")

WizHook.Key = 51488399
MsgBox WizHook.GlobalProcExists(strName)
```

Hinweise
Der Name der Prozedur wird nicht auf Gültigkeit geprüft. Wenn im Name unzulässige Sonderzeichen vorkommen liefert die Funktion einfach „False" zurück.

5.41 HideDates

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Function HideDates() As Boolean

Argumente
- - -

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.42 IsMatchToDbcConnectString

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Prüft, ob der übergebene String dem Connection-String der Datenbank entspricht

Deklaration
Property IsMatchToDbcConnectString(bstrConnectionString As String) As Boolean

Argumente
bstrConnectionString zu überprüfender Connction-String

Rückgabewert
True, wenn der übergebene String dem Connection-String der Datenbank entspricht.
False, wenn der übergebene String nicht auf die aktuelle Datenbank verweist.

Code-Beispiel
```
WizHook.Key = 51488399
MsgBox WizHook.IsMatchToDbcConnectString(CurrentProject.Connection)
```

Hinweise
- - -

5.43 IsMemberSafe

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Function IsMemberSafe(dispid As Long) As Boolean

Argumente
dispid ???

Rückgabewert
???

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.44 IsValidIdent

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Prüft, ob der Name einer Variable gültig ist / den Regeln entspricht

Deklaration
Function IsValidIdent(Identifier As String) As Boolean

Argumente

Identifier	Name der zu überprüfenden Variable

Rückgabewert
True, wenn der Name den Regeln entspricht
False, wenn der Name nicht als Variable gültig ist

Code-Beispiel

```
Dim strName As String

strName = Nz(Me!txtName, "")

WizHook.Key = 51488399
MsgBox WizHook.IsValidIdent(strName)
```

Hinweise
- - -

5.45 Key

Access-Versionen
Diese Eigenschaft ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Eigenschaft, die das WizHook-Objekt aktiviert.

Deklaration
Property Key As Long

Argumente
Key 51488399 ist der notwendige Wert für die Aktivierung

Code-Beispiel
```
WizHook.Key = 51488399

'Die folgende Zeile funktioniert nicht
'MsgBox WizHook.Key
'Diese Property ist Write-Only
```

Hinweise
Diese Eigenschaft kann nicht gelesen werden. Sie ist write-only.

5.46 KeyboardLangID

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Ermittelt die aktuell für die Tastatur eingestellte Sprach-ID.

Deklaration
Function KeyboardLangID() As Long

Argumente
- - -

Rückgabewert
1031 für Deutschland

Code-Beispiel
```
WizHook.Key = 51488399
MsgBox WizHook.KeyboardLangID
```

Hinweise
- - -

5.47 KnownWizLeaks

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Sub KnownWizLeaks(fStart As Boolean)

Argumente
fStart ???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.48 LoadImexSpecSolution

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
Sub LoadImexSpecSolution(bstrFilename As String)

Argumente
bstrFilename ???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Wenn als Argument eine Datei übergeben wird, die nicht existiert, gibt es eine Fehlermeldung.

Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.49 LocalFont

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Ermittelt den standardmäßig von der Access verwendeten Zeichensatz.

Deklaration
Function LocalFont() As String

Argumente
- - -

Rückgabewert
Name des verwendeten Fonts, z.B. „Tahoma"

Code-Beispiel
```
WizHook.Key = 51488399
MsgBox WizHook.LocalFont
```

Hinweise
- - -

5.50 NameFromActid

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Liefert den Namen einer Access VBA-Aktion zur Action-ID

Deklaration
Function NameFromActid(Actid As Long) As String

Argumente
Actid Zahlenwert, der die Action-ID angibt

Rückgabewert
Name der Access VBA-Aktion

Code-Beispiel
```
Dim lngActID As Long

lngActID = CLng(Nz(Me!txtActID, 0))

WizHook.Key = 51488399
MsgBox "Action-ID: " & lngActID & vbCrLf & _
       "Name : " & WizHook.NameFromActid(lngActID) & vbCrLf & _
       "Argumente: " & WizHook.ArgsOfActid(lngActID)
```

Hinweise
Siehe auch „ArgsOfActid", „GetScriptString" und „Action-ID's und ihre Aktionen".

Als Action-ID können Werte zwischen 1 und 65 eingegeben werden. Alle anderen Werte liefern „0" zurück.

5.51 ObjTypOfRecordSource

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Ermittelt den Typ einer Datenquelle

Deklaration
Function ObjTypOfRecordSource(RecordSource As String) As Integer

Argumente
RecordSource Datenquelle

Rückgabewert
0 SQL-Ausdruck
1 Tabelle
2 Abfrage
-1 Das Objekt existiert nicht

Code-Beispiel
```
Dim strName As String

strName = Nz(Me!txtName, "")

WizHook.Key = 51488399
MsgBox WizHook.ObjTypOfRecordSource(strName)
```

Hinweise
Wenn ein SQL-Ausdruck auf ein Objekt verweist, das nicht existiert, liefert die Funktion trotzdem „0" zurück.

5.52 OfficeAddInDir

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Ermittelt das Office-Addin-Verzeichnis

Deklaration
Function OfficeAddInDir() As String

Argumente
- - -

Rückgabewert
Pfad zu den Office-AddIns

Code-Beispiel
```
WizHook.Key = 51488399
MsgBox WizHook.OfficeAddInDir
```

Hinweise
- - -

5.53 OpenEmScript

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck

????

Deklaration

```
Function OpenEmScript(pProperty As _AccessProperty,
                      OpenMode As Long,
                      Extra As Long,
                      Version As Long) As Long
```

Argumente

pProperty	???
OpenMode	???
Extra	???
Version	???

Rückgabewert

???

Code-Beispiel

- - -

Hinweise

Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.54 OpenPictureFile

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Öffnet einen Dateiauswahldialog mit dem Titel „Grafik einfügen"

Deklaration
Function OpenPictureFile(File As String,
Cancelled As Boolean) As Boolean

Argumente
File	Rückgabewert der Funktion. Enthält den ausgewählten Dateinamen.
Cancelled	Rückgabewert der Funktion. Ist True, wenn der Dialog abgebrochen wurde.

Rückgabewert
True, wenn der Aufruf erfolgreich war.
False, wenn der Aufruf der Funktion fehlgeschlagen ist.

Code-Beispiel
```
Dim ret As Boolean
Dim strFile As String
Dim fCancelled As Boolean

WizHook.Key = 51488399
ret = WizHook.OpenPictureFile(strFile, fCancelled)
If ret = True Then
    If fCancelled = False Then
        MsgBox strFile
    End If
End If
```

Hinweise
Bei dieser Funktion hat es sich um eine spezielle Form des GetFileName-Dialogs.

Der Nachteil ist, dass diese Funktion nicht konfiguriert werden kann.

5.55 OpenScript

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Öffnet ein Makro und liefert einen Verweis darauf zurück.

Deklaration
```
Function OpenScript(Script As String,
                    Label As String,
                    OpenMode As Long,
                    Extra As Long,
                    Version As Long) As Long
```

Argumente
Script	Name des Makros (wie im Datenbankfenster eingetragen)
Label	Name des Makros (wie in der Spalte „Makroname" eingetragen)
OpenMode	Öffnungsmodus: 0: Lesen 2: Schreiben
Extra	???
Version	Version des Makros

Rückgabewert
Verweis auf das Makro.
Dieser wird von den Funktionen „GetScriptString" und „SaveScriptString" benötigt.

Code-Beispiel
```
Dim hScr As Long
Dim strScript As String
Dim strLabel As String
Dim lngOpenMode As Long
Dim lngExtra As Long
Dim lngVersion As Long

'Initialisieren
strScript = Nz(Me!txtMakroName, "")
lngOpenMode = 0

'Makro öffnen
WizHook.Key = 51488399
hScr = WizHook.OpenScript(strScript, strLabel, lngOpenMode, lngExtra, lngVersion)

'Info ausgeben
If hScr > 0 Then
    MsgBox "Handle: " & hScr & vbCrLf & _
           "Extra: " & lngExtra & vbCrLf & _
           "Version: " & lngVersion
    'Makro wieder schliessen
    fCloseHscr hScr
Else
    MsgBox "Markro mit dem Namen " & strScript & vbCrLf & _
           "konnte nicht geöffnet werden."
End If
```

```
'Zusätzlich muss folgende Funktion deklariert sein:
Private Declare Sub fCloseHscr _
        Lib "msaccess.exe" _
        Alias "#20" _
        (ByVal hScr As Long)
```

Hinweise

Siehe auch „GetScriptString" und „SaveScriptString".

Wenn ein Makro im Modus „Schreiben" geöffnet wird, wird ein neues Makro mit dem gewünschten Namen angelegt. Wenn dieses Makro bereits besteht, wird ein neues, leeres Makros mit dem gleichen Namen angelegt.

Wenn an die Funktion „fCloseHscr" ein nicht existierendes Makro-Handle übergeben wird kommt es bei Access 2002 zu einem Absturz.

5.56 ReportLeaksToFile

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
```
Sub ReportLeaksToFile(fRptToFile As Boolean,
                      bstrFileOut As String)
```

Argumente
fRptToFile	???
bstrFileOut	???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.57 SaveObject

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Speichert ein Datenbankobjekt

Deklaration

Sub SaveObject(bstrName As String,
objtyp As Integer)

Argumente

bstrName	Name des zu speichernden Access-Objektes
objtyp	Typ des Access-Objektes (z.B. acForm) wie in der Enumeration „AcObjectType" aufgezählt.

Rückgabewert

- - -

Code-Beispiel

```
Dim strSeite As String

strSeite = Nz(Me!txtSeite, "")

WizHook.Key = 51488399
WizHook.SaveObject strSeite, acDataAccessPage
```

Hinweise

Das Objekt wird ohne weitere Nachfrage gespeichert.

Mit dem Befehl DoCmd.Save kann man denselben Erfolg erzielen.

5.58 SaveScriptString

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Schreibt einen Wert in eine Spalte eines Makros.

Deklaration

```
Function SaveScriptString(HScr As Long,
                          ScriptColumn As Long,
                          Value As String) As Boolean
```

Argumente

HScr	Referenz auf ein geöffnetes Makro
ScriptColumn	Zahl, die die Spalte angibt 0: Name des Markos 1: Kommentar 2: Bedingung 3-12: Argumente
Value	Wert, der eingetragen werden soll

Rückgabewert

True, wenn der Wert geschrieben werden konnte.
False, wenn beim Schreiben des Wertes ein Fehler aufgetreten ist.

Code-Beispiel

```
Dim hScr As Long
Dim strScript As String
Dim strLabel As String
Dim lngOpenMode As Long
Dim lngExtra As Long
Dim lngVersion As Long

'Initialisieren
strScript = "mak_MakroTest"
strLabel = ""
lngOpenMode = 2

'Makro öffnen
WizHook.Key = 51488399
hScr = WizHook.OpenScript(strScript, strLabel, lngOpenMode, lngExtra, lngVersion)

'Ausstieg bei Fehler
If hScr = 0 Then
    MsgBox "Markro konnte nicht geöffnet werden."
    Exit Sub
End If

'Werte schreiben
api_Makro_NextRow hScr, 0&, 0&
fSaveActidHscr hScr, 22&
WizHook.SaveScriptString hScr, 0&, "Name des Makros"
WizHook.SaveScriptString hScr, 1&, "Kommentar"
WizHook.SaveScriptString hScr, 2&, "..."
WizHook.SaveScriptString hScr, 3&, "Text für MsgBox"
```

```
WizHook.SaveScriptString hScr, 4&, True
WizHook.SaveScriptString hScr, 5&, 0
WizHook.SaveScriptString hScr, 6&, "Titel"

'Makro schliessen
fCloseHscr hScr

'Zusätzlich müssen folgende Funktionen deklariert sein:

Private Declare Function api_Makro_NextRow _
        Lib "msaccess.exe" _
        Alias "#22" _
        (ByVal Makro As Long, _
        ByVal lngSkipBlank As Long, _
        lngEndOfMakro As Long) As Long
Private Declare Function fSaveActidHscr _
        Lib "msaccess.exe" _
        Alias "#25" _
        (ByVal hScr As Long, _
        ByVal actid As Long) As Long
Private Declare Sub fCloseHscr _
        Lib "msaccess.exe" _
        Alias "#20" _
        (ByVal hScr As Long)
```

Hinweise

Siehe auch „OpenScript", „GetScriptString" und „NameFromActid".

Damit die vorgenommenen Änderungen endgültig gespeichert werden muss das Makro geschlossen werden.

5.59 SetDefaultSpecName

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2007, 2010

Funktion/Zweck
????

Deklaration
Sub SetDefaultSpecName(bstrSpecName As String)

Argumente
bstrSpecName ???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.60 SetDpBlockKeyInput

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Blockiert die Tastatureingaben für eine Datenzugriffs-Seite.

Deklaration
Sub SetDpBlockKeyInput(fBlockKeys As Boolean)

Argumente
fBlockKeys	True, Wenn die Tastatureingaben blockiert werden sollen. False, um die Tastatureingaben wieder zuzulassen.

Rückgabewert
- - -

Code-Beispiel
```
DoCmd.OpenDataAccessPage "Seite1"

WizHook.Key = 51488399
WizHook.SetDpBlockKeyInput True
```

Hinweise
Wenn bei der Ausführung des Codes das aktuelle Datenbankobjekt keine Datenzugriffsseite ist kommt es zu einem Automatisierungsfehler.

Die Blockierung der Tastatureingabe gilt nur für die bei der Ausführung des Befehls offenen Datenzugriffsseiten. Andere Datenzugriffsseiten sind davon nicht betroffen.

5.61 SetVbaPassword

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Setzt das VBA-Passwort

Deklaration
Function SetVbaPassword(bstrDbName As String,
bstrConnect As String,
bstrPasswd As String) As Boolean

Argumente

bstrDbName	Pfad und Name der Datenbank
bstrConnect	???
bstrPasswd	VBA-Passwort

Rückgabewert
True, wenn die Aktion erfolgreich.
False, im Falle des Misserfolgs.

Code-Beispiel
```
Dim strDBName As String
Dim strConnect As String
Dim strVBAPWD As String

'Daten übernehmen
strDBName = "C:\MeinPfad\MeineDB.mdb"
strConnect = ""
strVBAPWD = "MeinPasswort"

'VBA-Passwort ändern
WizHook.Key = 51488399
If WizHook.SetVbaPassword(strDBName, strConnect, strVBAPWD) = True Then
    MsgBox "VBA-Password gesetzt!"
Else
    MsgBox "Aktion fehlgeschlagen!"
End If
```

Hinweise
Wenn diese Funktion auf die gerade geöffnete Datenbank angewendet wird, meldet die Funktion zwar die erfolgreiche Ausführung. Das VBA-Passwort wird jedoch nicht gesetzt.
Wenn die Datenbank bereits ein VBA-Passwort besitzt, und das bisherige VBA-Passwort erneut gesetzt wird, verläuft die Funktion erfolgreich. Wenn jedoch ein anderes VBA-Passwort gesetzt werden soll gibt es eine Fehlermeldung. Ein bestehendes VBA-Passwort kann also nicht geändert werden.
Dieser Code kann nicht auf ein Access-Projekt (*.adp) angewendet werden.

5.62 SetWizGlob

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
????

Deklaration
```
Sub SetWizGlob(lWhich As Long,
               vValue)
```

Argumente

lWhich	???
vValue	???

Rückgabewert
- - -

Code-Beispiel
- - -

Hinweise
Es ist mir bisher noch nicht gelungen, diese Funktion zu dokumentieren.

5.63 SortStringArray

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Sortiert ein Array von Textfeldern.

Deklaration
Sub SortStringArray(Array() As String)

Argumente

Array()	Array mit Texten

Rückgabewert
- - -

Code-Beispiel

```
Dim strArray(3) As String

'Werte setzen
strArray(0) = "Delta"
strArray(1) = "Bravo"
strArray(2) = "Alpha"
strArray(3) = "Charly"

'Ausgabe
MsgBox strArray(0) & " / " & strArray(1) & " / " & strArray(2) & " / " & _
    strArray(3), , "Vorher:"

'Sortieren
WizHook.SortStringArray strArray()

'Ausgabe
MsgBox strArray(0) & " / " & strArray(1) & " / " & strArray(2) & " / " & _
    strArray(3), , "Nachher:"
```

Hinweise
Diese Funktion läuft, ohne dass der Wert für den „Key" gesetzt werden muss.

Wenn einem Feld des Array noch kein Wert zugewiesen wurde, stürzt Access ab.

Das Sortieren erfolgt Case-Sensitiv. Es wird Kleinschreibung vor Großschreibung sortiert.

5.64 SplitPath

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Zerlegt einen Dateipfad in die Bestandteile Laufwerk, Verzeichnis, Dateiname und -endung

Deklaration
Sub SplitPath(Path As String,
Drive As String,
Dir As String,
File As String,
Ext As String)

Argumente

Path	Eingabe: Dateipfad
Drive	Ausgabe: Laufwerk
Dir	Ausgabe: Verzeichnis
File	Ausgabe: Dateiname
Ext	Ausgabe: Dateiendung

Rückgabewert
- - -

Code-Beispiel
```
Dim strDatei As String
Dim strLW As String
Dim strPfad As String
Dim strDateiName As String
Dim strSuffix As String

'Werte übernehmen
strDatei = SysCmd(acSysCmdAccessDir) & "MSACCESS.EXE"

'Pfad zerlegen
WizHook.Key = 51488399
WizHook.SplitPath strDatei, strLW, strPfad, strDateiName, strSuffix

'Ausgabe
MsgBox "Eingabe: " & strDatei & vbCrLf & vbCrLf & _
       "Ausgabe: " & vbCrLf & _
       "Laufwerk: " & strLW & vbCrLf & _
       "    Pfad: " & strPfad & vbCrLf & _
       "   Datei: " & strDateiName & vbCrLf & _
       "  Suffix: " & strSuffix
```

Hinweise
Die Funktion ist unabhängig von der Existenz des übergebenen Dateipfades.

Die Funktion ist sehr robust. Wenn nur ein Teil des Dateipfades übergeben wird nur dieser zerlegt.

5.65 TableFieldHasUniqueIndex

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck
Prüft, ob eine Spalte in einer Tabelle einen eindeutigen Index hat.

Deklaration
Function TableFieldHasUniqueIndex(Table As String,
Columns As String) As Boolean

Argumente
Table	Name der Tabelle
Columns	Name der Spalte bzw. des Feldes

Rückgabewert
True, wenn die Spalte in der Tabelle einen eindeutigen Index hat.
False, wenn die Spalte in der Tabelle keinen eindeutigen Index hat.

Code-Beispiel
```
'Variablen deklarieren
Dim strTabelle As String
Dim strFeld As String

'Werte übernehmen
strTabelle = „Tabellenname“
strFeld = „Feldname“

WizHook.Key = 51488399
If WizHook.TableFieldHasUniqueIndex(strTabelle, strFeld) = True Then
    MsgBox "Eindeutiger Index vorhanden."
Else
    MsgBox "Kein eindeutiger Index vorhanden."
End If
```

Hinweise
Wenn die Tabelle nicht vorhanden ist, gibt die Funktion „False“ zurück.
Wenn das Feld nicht vorhanden ist, gibt die Funktion ebenfalls „False“ zurück“.

Das zu übergebende Argument lautet „Columns“, also Spalten. Es ist mir jedoch nicht gelungen, diese Funktion mit mehreren Spalten, die zusammen einen gemeinsamen eindeutigen Index bilden, anzuwenden.

5.66 TranslateExpression

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Wandelte eine übergegeben Zeichenfolge in einen gültigen Ausdruck um. Dabei wird sämtlicher „Whitespace" entfernt.

Deklaration

```
Function TranslateExpression(In As String,
                             Out As String,
                             ParseFlags As Long,
                             TranslateFlags As Long) As Boolean
```

Argumente

In	umzuwandelnde Zeichenfolge
Out	Ergebnis der Umwandlung
ParseFlags	16: Alle Objektbezeichnungen werden in eckige Klammern gesetzt. 64: Führt bei mir zum Absturz von Access.
TranslateFlags	???

Rückgabewert

True, wenn die Ausführung erfolgreich verlaufen ist.
False, wenn bei der Ausführung Fehler aufgetreten sind.

Code-Beispiel

```
Dim strIn As String
Dim strOut As String
Dim lngParseFlags As Long
Dim lngTranslateFlags As Long

strIn = " 7 + Me.MeinFeld"
strOut = ""
lngParseFlags = 0
lngTranslateFlags = 0

WizHook.Key = 51488399
If WizHook.TranslateExpression(strIn, strOut, lngParseFlags, lngTranslateFlags) _
        = True Then
    MsgBox strOut
Else
    MsgBox "Aktion fehlgeschlagen!"
End If
```

Hinweise

Bei meinen Test war es egal, welcher Wert für TranslateFlags eingegeben wurde. Ich konnte keine Auswirkungen feststellen.

5.67 TwipsFromFont

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, 2007, 2010

Funktion/Zweck

Mit dieser Funktion ist es möglich, die Breite und Höhe eines bestimmten Textes mit bestimmten Textattributen zu ermitteln. Das Ergebnis wird in Twips angegeben.

Deklaration

```
Function TwipsFromFont(FontName As String,
                       Size As Long,
                       Weight As Long,
                       Italic As Boolean,
                       Underline As Boolean,
                       Cch As Long,
                       Caption As String,
                       MaxWidthCch As Long,
                       dx As Long,
                       dy As Long) As Boolean
```

Argumente

FontName	Name des Zeichensatzes
Size	Schriftgröße in Punkten
Weight	Schriftbreite: 400 = Normal; 700 = Fett weitere Werte finden sich in der OH zur FontWeight-Eigenschaft (Schriftbreite)
Italic	True = Kursiv; False = nicht kursiv
Underline	True = unterstrichen; False = nicht unterstrichen
Cch	Anzahl an Zeichen mit durchschnittlicher Breite
Caption	Text, für den die Maße bestimmt werden sollen
MaxWidthCch	Anzahl an Zeichen mit maximaler Breite
dx	Ermittelte Breite des Textes in Twips
dy	Ermittelte Höhe des Textes in Twips

Rückgabewert

True, wenn die Berechnung erfolgen konnte.
False, wenn bei der Berechnung Fehler aufgetreten sind.

Code-Beispiel

```
'Variablen deklarieren
Dim strCaption As String
Dim strFontName As String
Dim lngSGroesse As Long
Dim lngSBreite As Long
Dim fItalic As Boolean
Dim fUnderline As Boolean
Dim lngBreite As Long
Dim lngHoehe As Long

'Werte übernehmen
strCaption = "Dies ist ein Beispiel"
strFontName = "Arial"
lngSGroesse = 11
lngSBreite = 400
```

```
fItalic = False
fUnderline = False

Wizhook.Key = 51488399
If Wizhook.TwipsFromFont(strFontName, lngSGroesse, lngSBreite, fItalic, _
           fUnderline, 0, strCaption, 0, lngBreite, lngHoehe) = True Then
    MsgBox "Der genannte Text hat mit den gewählten" & vbCrLf & _
           "Schriftattributen folgende Ausmaße:" & vbCrLf & vbCrLf & _
           "Breite: " & lngBreite & vbCrLf & _
           "Höhe: " & lngHoehe
Else
    MsgBox "Berechnung fehlgeschlagen", vbExclamation
End If
```

Hinweise

Im Normalfall wird der Text aus dem Parameter „Caption" für die Berechnung herangezogen.

Wenn für den Parameter „Cch" ein Wert ungleich 0 angegeben wird, wird der Text im Parameter „Caption" ignoriert. Stattdessen wird die Anzahl aus dem Parameter „Cch" als Zeichen mit durchschnittlicher Breite für die Berechnung berücksichtigt.

Der im Parameter „MaxWidthCch" angegebene Wert wird nur berücksichtigt, wenn der Parameter „Cch" einen Wert ungleich 0 hat. In diesem Fall wird die Anzahl aus dem Parameter „MaxWidthCch" als Zeichen mit maximaler Breite für die Berechnung berücksichtigt. Die Gesamtzahl der Zeichen steht in diesem Fall im Parameter „Cch". Es werden dann „Cch" – „MaxWidthCch" Zeichen mit durchschnittlicher Breite und „MaxWidthCch" Zeichen mit maximaler Breite berücksichtigt. Wenn „MaxWidthCch" größer ist als „Cch", dann werden „Cch" Zeichen mit maximaler Breite berücksichtigt.

Man kann diese Funktion beispielsweise in folgenden Fällen einsetzen:

- In einem Listenfeld soll eine Textzeile mit variablem Inhalt als Überschrift unterstrichen werden. Mit der Funktion kann die Anzahl der notwendigen Zeichen für die Unterstreichung ermittelt werden.
- Ein Kombinationsfeld mit „unbekannten" Werten sollte so gestaltet werden, dass der längste Eintrag gerade noch vollständig sichtbar ist. Mit der Funktion kann die notwendige Listenbreite ermittelt werden.

5.68 WizCopyCmdbars

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
Importiert die benutzerdefinierten Befehlsleisten aus den angegebenen Datenbank.

Deklaration
Sub WizCopyCmdbars(bstrADPName As String)

Argumente
bstrADPName Name und Pfad zur Quelldatenbank

Rückgabewert
- - -

Code-Beispiel
```
Dim strDB As String

strDB = Nz(Me!txtPfad) & Nz(Me!txtNameDB)

WizHook.Key = 51488399
WizHook.WizCopyCmdbars strDB
```

Hinweise
Vorsicht: Wenn die Menü- oder Symbolleiste in der importierenden Datenbank ergänzt wurde und in der Quelldatenbank unverändert geblieben ist, dann werden beim nochmaligen importieren die Änderungen aus der Zieldatenbank in die Quelldatenbank zurück geschrieben.

Ich danke Michel Fouquet für den entscheidenden Hinweis.

5.69 WizHelp

Access-Versionen

Diese Funktion ist in folgenden Access-Versionen verfügbar: 2000, 2002, 2003, (2007) , 2010

Funktion/Zweck

Es kann eine Hilfedatei an einer bestimmten Stelle geöffnet werden.

Deklaration

```
Function WizHelp(HelpFile As String,
                 wCmd As Long,
                 ContextID As Long) As Boolean
```

Argumente

HelpFile	Pfad zur Hilfedatei
wCmd	? (Konstante für Befehl? Siehe unter Hinweise)
ContextID	Hilfekontextkennung

Rückgabewert

Bei all meinen Tests wurde als Rückgabewert „Wahr" zurückgegeben. Dies geschah unabhängig von den übergebenen Argumenten.

Code-Beispiel

```
Dim strHelpFile As String
Dim lngCmd As Long
Dim lngHelpID As Long

'Werte übernehmen
strHelpFile = "C:\Eigene Dateien\MeinHelpFile.chm"
lngCmd = 1
lngHelpID = 5

Wizhook.Key = 51488399
MsgBox Wizhook.WizHelp(strHelpFile, lngCmd, lngHelpID)
```

Hinweise

Ich habe für das Argument „wCmd" systematisch die Werte von 0 bis 32 probiert:

- 1 oder 11: Diese Werte sorgen dafür, dass die Hilfedatei an der gewünschten Stelle geöffnet wird.
- 9: Mit diesem Wert wird die Windows-Hilfe geladen, die Hilfedatei aber nicht.
- 10 und 12: Diese Werte führen zu einem Absturz von Access.
- Alle anderen: Bei allen anderen Werten passiert nichts.

In Access 2007 ist diese Funktion (noch) vorhanden, sie scheint aber nicht mehr zu funktionieren.

5.70 WizMsgBox

Access-Versionen
Diese Funktion ist in folgenden Access-Versionen verfügbar: 2002, 2003, 2007, 2010

Funktion/Zweck
Es wird eine Msgbox dargestellt.

Deklaration
```
Function WizMsgBox(bstrText As String,
                   bstrCaption As String,
                   wStyle As Long,
                   idHelpID As Long,
                   bstrHelpFileName As String) As Long
```

Argumente

bstrText	Text, der in der MsgBox erscheint
bstrCaption	Überschrift
wStyle	Konstanten, wie sie von der MsgBox-Funktion bekannt sind (wie in der Enumeartion „VbMsgBoxStyle" aufgezählt.)
idHelpID	Hilfekontextkennung
bstrHelpFileName	Pfad zur Hilfedatei

Rückgabewert
Es werden Werte, wie sie von der MsgBox-Funktion her bekannt sind, zurückgegeben (wie in der Enumeration „VbMsgBoxResult" aufgezählt.)

Code-Beispiel
```
Dim strText As String
Dim strTitel As String
Dim strHelpFile As String

strText = "Dies ist der Text." & vbCrLf & _
          "Dies ist die zweite Zeile." & vbCrLf & _
          "Hier kommt die Dritte."
strTitel = "Hier steht die Überschrift"
strHelpFile = "C:\MeinHelpFile.chm"
WizHook.Key = 51488399

If WizHook.WizMsgBox(strText, strTitel, vbYesNo, 1, strHelpFile) = vbYes Then
    MsgBox "Es wurde Ja gedrückt."
End If
```

Hinweise
Die Hilfe-Schaltfläche wird dargestellt, wenn für den Parameter „idHelpID" ein Wert größer 0 angegeben wird.

Diese Funktion ist im Wesentlichen gleich zur bekannten MsgBox-Funktion. Mit der MsgBox-Funktion gelingt es mir jedoch nicht, eine Hilfedatei zu öffnen.
Hierin liegt der Vorteil dieser Funktion. Mit ihr kann ich eine Hilfedatei an der gewünschten Stelle öffnen.

6 Anregungen und Ergänzungen

Dieses Dokument lebt. Es ist Stück für Stück entstanden – und es ist sicher noch nicht vollständig.

Anregungen und Ergänzungen zu diesem Dokument sind immer willkommen. Am besten senden Sie mir ein E-Mail an Access@Team-Moeller.de.

Auf diesem Weg kann das Wissen um das WizHook-Objekt in Access nach und nach ergänzt und vervollständigt werden.

7 Links und andere Quellen

Wenn man mit dem Begriff „WizHook" bei www.google.de eine Suche startet, dann findet man nur ca. 50 Treffer. (Stand 05/2005)
Die meisten Treffer verweisen auf Webseiten in asiatischer oder russischer Sprache. Andere Webseiten sind schlicht nicht mehr verfügbar.
Es finden sich auch einige Treffer in Forumsdiskussionen. Dabei geht es meistens darum, dass es das WizHook-Objekt gibt. Inhaltliche Beschreibungen sind selten.

Hier ist eine Liste mit meinen bevorzugten Links:

- **Juan M. Afán de Ribera, spanischer MVP**
  Von diesem spanischen MVP wurde ca. die Hälfte der Funktionen des WizHook-Objekts mit Beispielen dokumentiert. Leider sind die Seiten in spanisch.
  http://www.mvp-access.es/juanmafan/wizhook/wizhook.htm
- **Jason M.**
  Hier wurden in englischer Sprache etwas mehr als die Hälfte der Eigenschaften und Methoden des WizHook-Objekts dokumentiert.
  http://ftp-developpez.com/cafeine/access/access_wizhook.pdf
- **Access 97 Developers Handbook**
  In Kapitel 19 beschreiben die Autoren einige Funktionen aus der Vorgängerversion.
  http://www.developershandbook.com/

8 Anhang

8.1 Action-ID's und ihre Aktionen

Die nachfolgende Übersicht hilft bei der Erstellung von Makros mit der Funktion "SaveScriptString":

Act-ID	Aktion	Anz. Args	Act-ID	Aktion	Anz. Args
1	AddMenu	3	34	RunMacro	3
2	- -		35	RunSQL	2
3	ApplyFilter	2	36	- -	
4	Beep	0	37	SelectObject	3
5	CancelEvent	0	38	SendKeys	2
6	Close	3	39	- -	
7	CopyObject	4	40	SetValue	2
8	DoMenuItem	4	41	SetWarnings	1
9	Echo	2	42	- -	
10	- -		43	ShowAllRecords	0
11	FindNext	0	44	StopAllMacros	0
12	FindRecord	7	45	StopMacro	0
13	GoToControl	1	46	OpenReport	5
14	GoToPage	3	47	TransferDatabase	7
15	GoToRecord	4	48	TransferSpreadsheet	6
16	- -		49	TransferText	7
17	Hourglass	1	50	- -	0
18	- -		51	OutputTo	7
19	Maximize	0	52	DeleteObject	2
20	Minimize	0	53	OpenModule	2
21	MoveSize	4	54	SendObject	10
22	MsgBox	4	55	ShowToolbar	2
23	OpenForm	6	56	Save	2
24	OpenQuery	3	57	SetMenuItem	4
25	OpenTable	3	58	RunCommand	1
26	PrintOut	6	59	OpenDataAccessPage	2
27	Quit	1	60	OpenView	3
28	Requery	1	61	OpenDiagram	1
29	RepaintObject	2	62	OpenStoredProcedure	3
30	Rename	3	63	TransferSQLDatabase	6
31	Restore	0	64	CopyDatabaseFile	3
32	RunApp	1	65	OpenFunction	3
33	RunCode	1			