

# Mutex und Access



## Verwendung eines "Mutex" in Access in Verbindung mit Inno Setup

**Keywords: Mutex, Access, Inno Setup**

Copyright © 2002 Christoph Jüngling

Stand: 25. August 2002

### Copyright-Hinweis

Die Informationen in diesem Dokument wurden mit dem Ziel veröffentlicht, dass sie allen Menschen frei und uneingeschränkt zur Verfügung stehen sollen. Es wird die Erlaubnis gegeben dieses Dokument zu kopieren, verteilen und/oder zu verändern unter den Bedingungen der *GNU Free Documentation License*, Version 1.1 oder einer späteren, von der *Free Software Foundation* veröffentlichten Version. Eine Kopie dieser Lizenz ist auf <http://www.gnu.org/licenses/fdl.html> zu finden, eine deutsche Übersetzung auf <http://home.vr-web.de/juergen.katins/ruby/buch/gfdlde.html>.

Ich empfehle einen gelegentlichen Besuch auf [www.Juengling-EDV.de](http://www.Juengling-EDV.de), wo evtl. eine aktualisierte Version dieses Dokuments bereitsteht.

### Warenzeichen-Hinweis

Soweit in diesem Dokument Markennamen, Warenzeichen oder ähnliche geschützte Namen erwähnt werden, geschieht dies ausschließlich zu beschreibenden Zwecken. Sämtliche Rechte daraus stehen den jeweiligen Inhabern zu.

## Inhaltsverzeichnis

<b>1 Vorbemerkungen.....</b>	<b>3</b>
1.1 Motivation.....	3
1.2 Betrifft.....	3
<b>2 Was ist ein Mutex?.....</b>	<b>3</b>
<b>3 Wie geht das nun in Access?.....</b>	<b>4</b>
3.1 Deklarationen für die API-Funktionen.....	4
3.2 Mutex erzeugen.....	5
3.3 Mutex freigeben.....	5
3.4 Aufruf in der Applikation.....	6
3.4.1 Beispiel für die Initialisierung des kritischen Pfades.....	6
3.4.2 Beispiel für die Beendigung des kritischen Pfades.....	6
<b>4 Inno Setup und IStool.....</b>	<b>6</b>
<b>5 Das Ergebnis.....</b>	<b>7</b>
5.1 Setup.....	7
5.2 Doppelter Start.....	8

## Abbildungsverzeichnis

<b>Abbildung 1: Setup Options in IStool.....</b>	<b>7</b>
<b>Abbildung 2: Fehlermeldung von Inno Setup.....</b>	<b>7</b>
<b>Abbildung 3: Eigene Fehlermeldung.....</b>	<b>8</b>

# 1 Vorbemerkungen

## 1.1 Motivation

Durch den *Inno-Setup-Vortrag* von Michael Reitz auf der 4. AEK 2001 [1] wurde mein Interesse an diesem Thema geweckt. Dort wurde so ganz am Rande die Möglichkeit eines „Mutex“ angerissen, dessen Namen man in *Inno Setup* eingeben könne, und mit dem die Installation eines Update in gewisser Weise zu beeinflussen sei. Leider fehlte ein konkretes Beispiel im Stil „Wie geht das in Access?“. Seitdem hat mich das Thema nicht mehr losgelassen.

Im Verlaufe meiner Recherchen kam noch die Idee dazu, einen doppelten Start der Applikation zu verhindern. Dies wurde durch den Beispielcode von *Bryan Stafford's vbVision-Seite* [2] bewirkt, allerdings gehe ich einen etwas anderen Weg als Bryan, um das zu erreichen.

## 1.2 Betrifft

- Microsoft Access [3]
- Inno Setup [4]
- ISTool [5]

Der untenstehende Code ist unter folgenden Kombinationen getestet:

- Access 97 SR-2 unter Windows NT 4 SP6, Access 2000 unter Windows 2000, Access XP (2002) unter Windows 2000
- Inno Setup 2.0.14, 2.0.16, 3.0.2-beta
- ISTool 1.1.11, 3.0.2.1

# 2 Was ist ein Mutex?

Der Begriff *Mutex* ist ein Kunstwort aus den englischen Wörtern *mutual* und *exclusion* (dt. „gegenseitiger Ausschluss“). Ein Mutex wird in Systemen benötigt, bei denen eine Ressource (z.B. Variable, Gerät u.a.) von verschiedenen Programmteilen abwechselnd benutzt werden soll. Der Programmteil, in dem die Benutzung der Ressource erfolgt, wird als *critical section* („kritischer Abschnitt“) bezeichnet. Ein Mutex wird dabei von jedem der Programmteile am Anfang eines kritischen Abschnitts belegt und am Ende wieder freigegeben. Dabei muss allerdings geprüft werden, ob die Anforderung des Mutex' auch erfolgreich war. Falls nicht, ist der Mutex bereits belegt und der Programmcode des kritischen Abschnittes darf nicht ausgeführt werden. So wird verhindert, dass mehrere Programmteile auf die Ressource zugreifen. [6]

Im konkreten Fall sollen mittels des Mutex' zwei Ziele verfolgt werden: Es soll sichergestellt werden, dass

1. die Access-Applikation nur einmal gestartet wird (dies betrifft *Access* allein)
2. die Access-MDB/MDE nicht installiert werden kann, während sie bereits gestartet ist (dies betrifft *Access* in Verbindung mit *Inno Setup*)

Um dies zu erreichen muss die Applikation den Mutex beim Starten erzeugen und beim Beenden wieder freigeben. Im Sinne der obigen Definition besteht der *kritische Abschnitt* also aus der kompletten Access-Applikation.

### 3 Wie geht das nun in Access?

Es werden die API-Funktionen „CreateMutexA“, „OpenMutexA“ und „ReleaseMutex“ verwendet. Zusätzlich ist aus organisatorischen Gründen noch "CloseHandle" erforderlich.

Als Bezeichnung für den Mutex muss ein eindeutiger String vergeben werden. Dafür benutze ich naheliegenderweise den Namen der Applikation, den ich in meinen Projekten in einem Modul global deklariere und sowieso für diverse Aufgaben verwende (z.B. Titel des Applikationsfensters, Titel von Message-Boxen, Dateinamen für Datenexport, ...). Man könnte natürlich auch den Dateinamen verwenden oder einen beliebigen anderen String. Das Problem ist generell, dass dieser String betriebssystemweit eindeutig sein sollte. Wie man das sicherstellen kann, habe ich noch nicht herausgefunden.

```
Public Const ApplicationName = "Meine Test-Applikation"
```

Diese Bezeichnung wird später noch in der Inno-Setup-Konfiguration benötigt. Weiterhin benötige ich eine Variable vom Typ *Long*, in der der Handle des Mutex' gespeichert werden kann (das ist für die spätere Freigabe wichtig) und deren Lebensdauer der Applikation entspricht:

```
Public ApplicationMutex As Long
```

#### 3.1 Deklarationen für die API-Funktionen

```
Private Const READ_CONTROL = &H20000
```

```
Private Type SECURITY_ATTRIBUTES  
    nLength As Long  
    lpSecurityDescriptor As Long  
    bInheritHandle As Long  
End Type
```

```
Private Declare Function CreateMutex Lib „kernel32“ _  
    Alias „CreateMutexA“ (lpMutexAttributes As SECURITY_ATTRIBUTES, _  
    ByVal bInitialOwner As Long, ByVal lpName As String) As Long
```

```
Private Declare Function OpenMutex Lib „kernel32“ _  
    Alias „OpenMutexA“ (ByVal dwDesiredAccess As Long, _  
    ByVal bInheritHandle As Long, ByVal lpName As String) As Long
```

```
Private Declare Function ReleaseMutex Lib „kernel32“ _
    (ByVal hMutex As Long) As Long

Private Declare Function CloseHandle Lib "kernel32" _
    (ByVal hObject As Long) As Long
```

### 3.2 Mutex erzeugen

```
Public Function MutexErzeugen() As Boolean

Dim TestMutex As Long
Dim SecAtt As SECURITY_ATTRIBUTES

'-----

With SecAtt
    .nLength = Len(SecAtt)
    .lpSecurityDescriptor = CLng(0)
    .bInheritHandle = CLng(0)
End With

' Erstmal versuchen, den Mutex zu öffnen
TestMutex = OpenMutex(READ_CONTROL, CLng(0), ApplicationName)

If TestMutex > 0 Then
    ' Mutex existiert, also läuft die Applikation bereits

    ' Zum Testen verwendeten Handle freigeben
    CloseHandle TestMutex

    ' Status an aufrufende Routine zurückmelden
    MutexErzeugen = False
Else
    ' Mutex existiert noch nicht

    ' Mutex als Eigentümer erzeugen und Handle in
    ' globaler Variablen speichern
    ApplicationMutex = CreateMutex(SecAtt, CLng(1), ApplicationName)

    ' Status an aufrufende Routine zurückmelden
    MutexErzeugen = (ApplicationMutex > 0)

End If

End Sub
```

### 3.3 Mutex freigeben

```
Public Sub MutexFreigeben()

If ApplicationMutex > 0 Then ' Haben wir einen Handle drauf?
    ' Mutex freigeben
    ReleaseMutex ApplicationMutex

    ' Handle freigeben
    CloseHandle ApplicationMutex
End Sub
```

```
        ' Handle-Merker zurücksetzen
        ApplicationMutex = 0
    End If

End Sub
```

### 3.4 Aufruf in der Applikation

Durch die Kapselung der Erzeugung und Freigabe des Mutex' in zwei VB-Subs reduziert sich die Angelegenheit in der Applikation auf jeweils einen Aufruf in der Initialisierungs- und der Beendigungsroutine, z.B. im Öffnen- und Schließen-Ereignis eines Formulars, das mit Sicherheit immer geöffnet ist. Zur Not nimmt man ein ausgeblendetes Formular.

#### 3.4.1 Beispiel für die Initialisierung des kritischen Pfades

```
' Hauptformular der Applikation
Private Sub Form_Open(Cancel As Integer)

    If MutexErzeugen() = False Then
        MsgBox "Die Applikation läuft bereits.", _
            vbExclamation, ApplicationName
        DoCmd.Quit acQuitSaveAll
        ' ... oder eine andere gewünschte Reaktion
    End If

    Initialisierungen

End Sub
```

#### 3.4.2 Beispiel für die Beendigung des kritischen Pfades

```
' Hauptformular der Applikation
Private Sub Form_Close()

    Aufräumarbeiten
    MutexFreigeben

End Sub
```

## 4 Inno Setup und IStool

*Inno Setup* speichert den Mutexnamen in der Scriptdatei wie folgt:

```
[Setup]
AppMutex=Meine Test-Applikation
```

In *IStool* findet sich die Eintragung unter „Project ▢ Setup Options ▢ Application“ im Feld „Application mutex“:

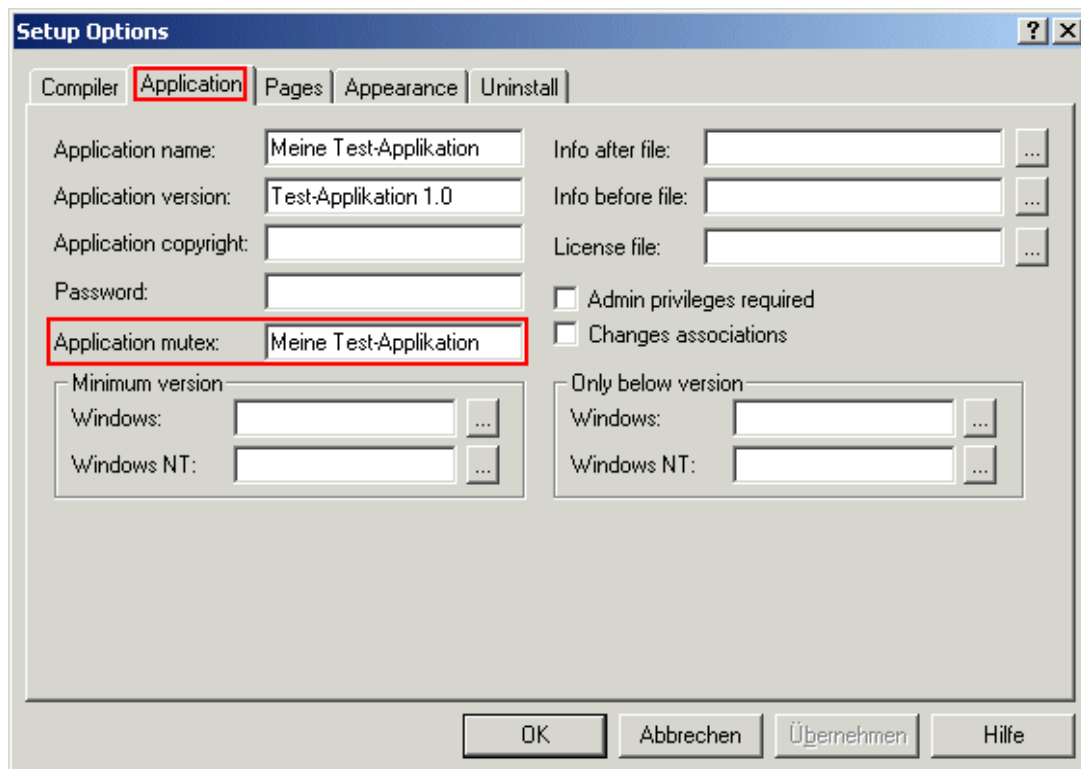


Abbildung 1: Setup Options in ISTool

Ob man nun mit ISTool (oder einer anderen Oberfläche für Inno Setup) oder dem Scripteditor arbeitet: Wichtig ist in beiden Fällen, dass man anstelle des hier gezeigten Strings „Meine Test-Applikation“ die gleiche Bezeichnung einträgt, mit der der Mutex in der Applikation erzeugt wurde.

## 5 Das Ergebnis

### 5.1 Setup

Wird versucht, während die Applikation läuft, ein Update zu installieren, so meldet sich das von *Inno Setup* generierte Setup z.B. mit folgender Message Box:

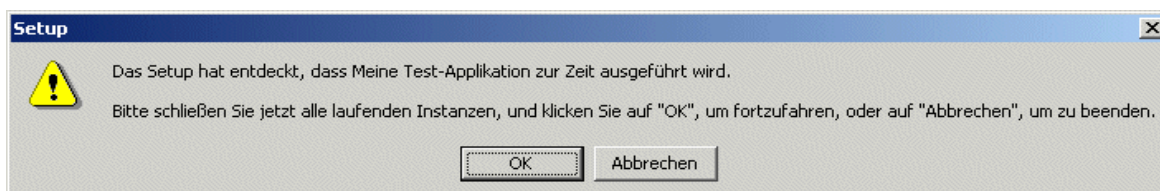


Abbildung 2: Fehlermeldung von Inno Setup

Erst wenn der Mutex freigegeben, also die Applikation beendet wurde, führt ein Klick auf „OK“ zu den weiteren Schritten des Setups, andernfalls kommt immer wieder diese Message-Box.

## 5.2 Doppelter Start

Wenn die Applikation beim Start entdeckt, dass eine Instanz bereits läuft, meldet Sie dies dem Benutzer z.B. mit folgender Message-Box und beendet sich daraufhin:

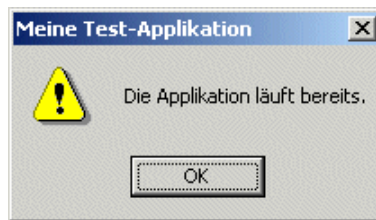


Abbildung 3: Eigene Fehlermeldung

Christoph Jüngling  
[www.juengling-edv.de](http://www.juengling-edv.de)

Wenn Sie dieses Dokument hilfreich fanden,  
würde ich mich über einen „Flattr“ sehr freuen.  
[Klicken Sie einfach hier.](#)

- 1 [] 4. AEK 2001, <http://www.donkarl.com/AEK/AEK4.htm>
- 2 [] Bryan Stafford: vbVision, <http://www.mvps.org/vbvision/>
- 3 [] <http://www.microsoft.com/office/access/>
- 4 [] <http://www.innosetup.org>, <http://www.jrsoftware.org>
- 5 [] <http://www.bhenden.org/istool/>
- 6 [] Definition nach Marko Meister, [http://www.uni-weimar.de/~meister1/private/papers/diplom/diplom\\_009.html](http://www.uni-weimar.de/~meister1/private/papers/diplom/diplom_009.html). Nach dessen Definition muss der Begriff „Mutex“ dem weiblichen Geschlecht zugeordnet werden. Rein gefühlsmäßig tendiere ich allerdings dazu, „Mutex“ als männlich zu betrachten.