Word + VBA

Shell-Funktion

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 / 365

Erstellungsdatum: 09.09.2022

Überarbeitung: 01.12.2023

Beschreibung: Die Shell-Funktion führt ein ausführbares Programm unter Windows aus und liefert einen Rückgabewert vom Typ Double. Dieser ist bei erfolgreicher Ausführung die Aufgaben-ID, andernfalls der Wert Null.

Anwendungs-Datei: AW-001_ShellFunktion.docm

1 Syntax

Die Shell-Funktion führt ein ausführbares Programm unter Windows aus und liefert einen Rückgabewert vom Typ *Double*. Dieser ist bei erfolgreicher Ausführung die Aufgaben-ID, andernfalls der Wert Null.

Die Syntax der Funktion lautet

Shell (Pfadname, [Fensterform]).

Tabelle 1. Die Parameter der Syntax

Parameter	Typ	Beschreibung
Pfadname	Variant	Name des auszuführenden Programms und alle erforderlichen Argumente. Er kann auch das Verzeichnis enthalten.
Fensterform	Variant	Legt den Stil der Fensterform fest (siehe Tabelle 2).

Tabelle 2. Fensterformen zur Shell-Funktion

Konstante	Wert	Beschreibung
vbHide	0	Das Fenster wird ausgeblendet, und der Fokus wird an das ausgeblendete Fenster übergeben.
vbNormalFocus	1	Das Fenster besitzt den Fokus und wird wieder in seiner ursprünglichen Größe und Position angezeigt.
vbMinimizedFocus	2	Das Fenster wird als Symbol mit dem Fokus angezeigt.
vbMaximizedFocus	3	Das Fenster ist maximiert und besitzt den Fokus.
vbNormalNoFocus	4	Das Fenster wird in seiner letzten Größe und Position wiederhergestellt. Das derzeit aktive Fenster bleibt aktiv.
vbMinimizedNoFocus	6	Das Fenster wird als Symbol angezeigt. Das derzeit aktive Fenster bleibt aktiv.

Kann die Shell-Funktion das angegebene Programm nicht starten, erfolgt eine Fehlermeldung. Standardmäßig führt die Shell-Funktion die aufgerufenen Programme asynchron aus, sodass ein Programm möglicherweise nicht abgeschlossen wird, bevor weitere Anweisungen der Shell-Funktion folgen.

2 Manuelle Ausführung

Die manuelle Ausführung von Programmen unter Windows erfolgt unter dem Kontextmenü (rechte Maustaste) des Windows-Startsymbols in der Tastleiste mit *Ausführen* (Bild 1).

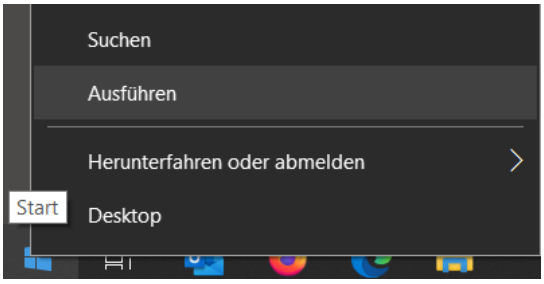

Bild 1. Ausführung von Programmen aus dem Kontextmenü des Startsymbols

In dem sich öffnenden Dialogfenster *Ausführen* wird der Name des Programms eingetragen und mit *OK* gestartet (Bild 2). Damit öffnet sich die Anwendung und erhält den Fokus (Bild 3).

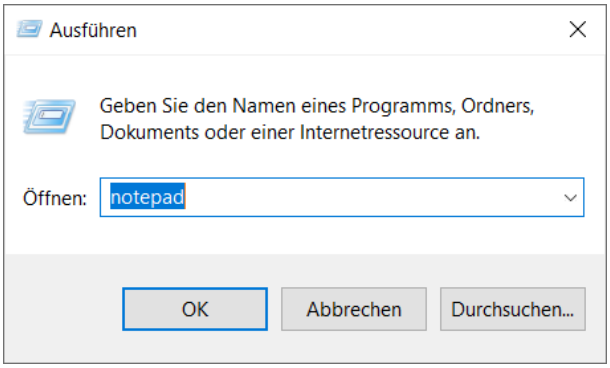

Bild 2. Aufruf der Anwendung Notepad

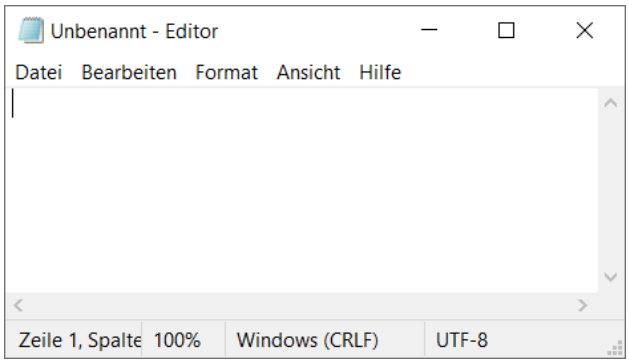

Bild 3. Der Texteditor Notepad

3 Start des Texteditors Notepad.exe

Mithilfe der Shell-Funktion lässt sich der Texteditor Notepad ebenfalls starten. Dazu verwenden wir eine einfache Testprozedur.

Codeliste 1. Die Testprozedur in ThisDocument startet den Texteditor Notepad

```
Option Explicit

Sub TestShellNotepad()
   Dim vReturn As Variant
   vReturn = Shell("NOTEPAD.EXE", 1)
End Sub
```

4 Start des Taschenrechners Calc.exe

Ein weiteres Hilfsprogramm wird ebenfalls gerne neben der Dokumentbearbeitung genutzt, es ist der Taschenrechner Calc.exe.

Codeliste 2. Die Testprozedur in ThisDocument startet den Taschenrechner Calc

```
Sub TestShellCalc()
   Dim vReturn As Variant
   vReturn = Shell("CALC.EXE", 1)
End Sub
```

Nach dem Start öffnet sich das Fenster des Rechners (Bild 4).

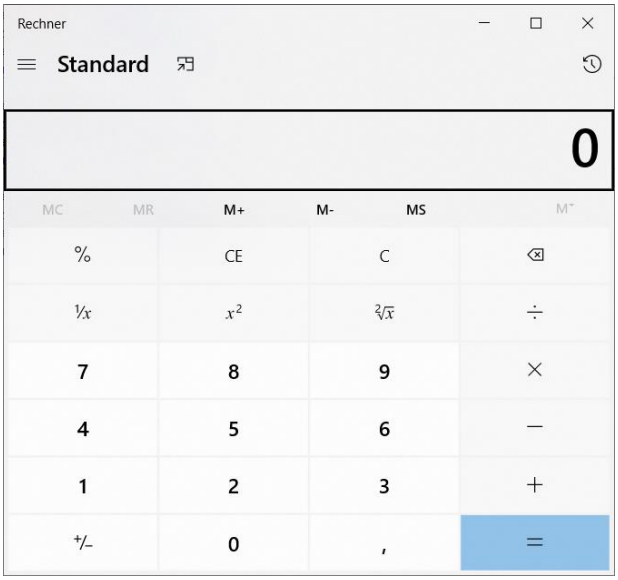

Bild 4. Das Dialogfenster des Rechners

Unter den Windows-Programmen sind einige sehr umständlich oder garnicht über die grafische Oberfläche zugänglich. Nachfolgend werden einige hilfreiche davon beschrieben.

5 Start des Clean-Managers

Mithilfe des Clean-Managers können Festplatten von temporären Dateien gesäubert werden.

Codeliste 3. Die Testprozedur in ThisDocument startet den Clean-Manager

```
Sub TestShellCleanMgr()
   Dim vReturn As Variant
   vReturn = Shell("CLEANMGR.EXE", 1)
End Sub
```

In einem Dialogfenster wird das entsprechende Laufwerk abgefragt (Bild 5).

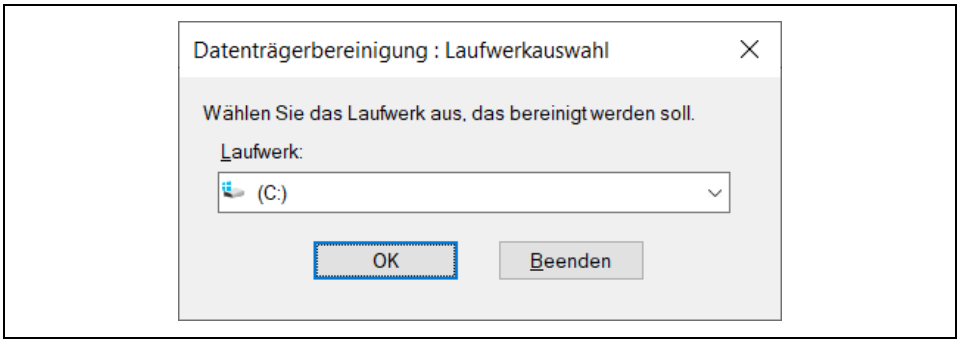

Bild 5. Laufwerksauswahl

Mit der Bestätigung durch *OK* erfolgt die Auswertung (Bild 6). In dem Dialogfenster werden verschiedene Vorschläge zur Datenbereinigung als Optionen angezeigt. Durch An- oder Abwahl einzelner Optionen wird die Vorgabe zur Datenbereinigung erstellt. Mit der Bestätigung durch *OK* erfolgt die Ausführung, deren Fertigstellung durch ein weiteres Dialogfenster gemeldet wird.

Eine zusätzliche Schaltfläche *Systemdateien bereinigen* kann zusätzlich gestartet werden und räumt unter den Systemdateien auf.

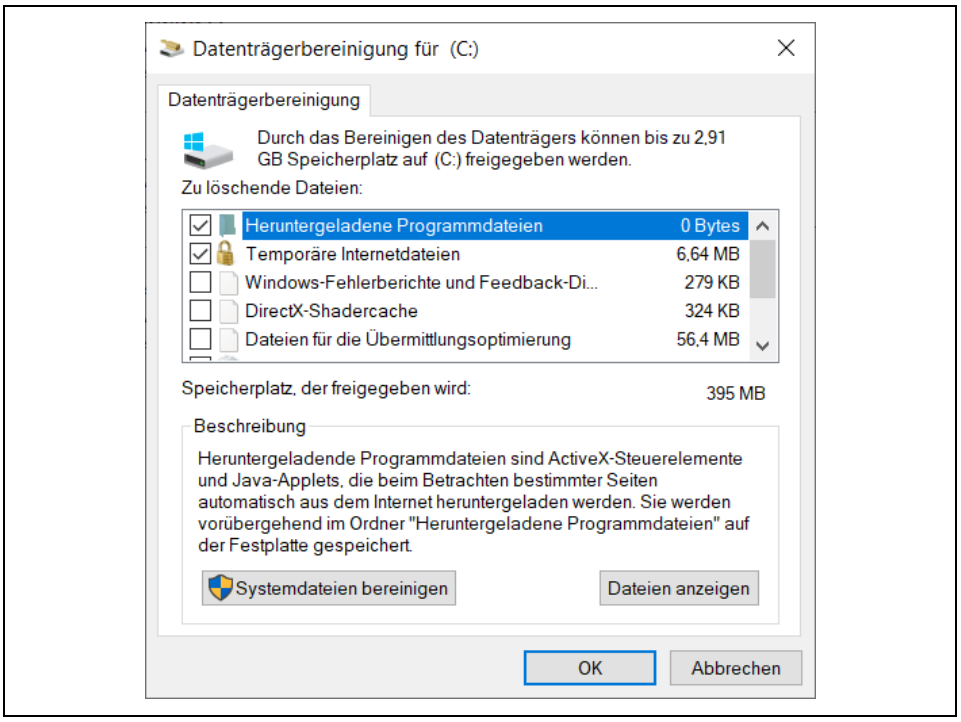

Bild 6. Dialogfenster zeigt Vorschläge zur Datenbereinigung

6 Start des DirectX-Diagnoseprogramms

Das Programm DxDiag liefert detaillierte Informationen über die DirectX-Komponenten und -Treiber des verwendeten Computers.

Codeliste 4. Die Testprozedur in ThisDocument startet das DirectX-Diagnoseprogramm

```
Sub TestShellDxDiag()
   Dim vReturn As Variant
   vReturn = Shell("DXDIAG.EXE", 1)
End Sub
```

Das Ergebnis der Diagnose wird in einem Dialogfenster mit mehreren Registern präsentiert (Bild 7).

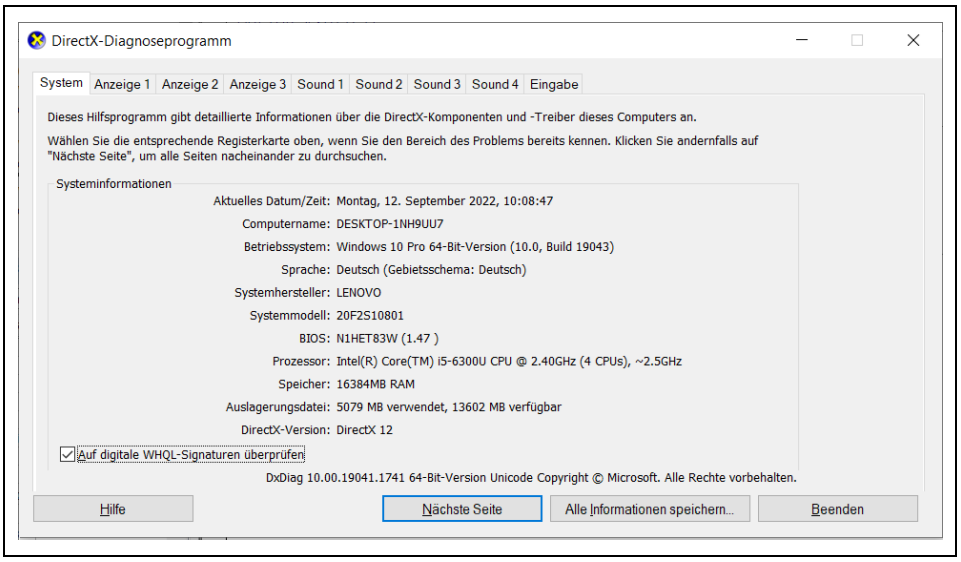

Bild 7. Ergebnisse der Diagnose

7 Start des Mobilisierungscenters

Das Programm *MobSync* dient zur Synchronisation verschiedener Objekte.

Codeliste 5. Die Testprozedur in ThisDocument startet das Synchronisationsprogramm MobSync

```
Sub TestShellMobSync()
   Dim vReturn As Variant
   vReturn = Shell("MOBSYNC.EXE", 1)
End Sub
```

Mit dem Start öffnet sich das Synchronisationscenter (Bild 8).

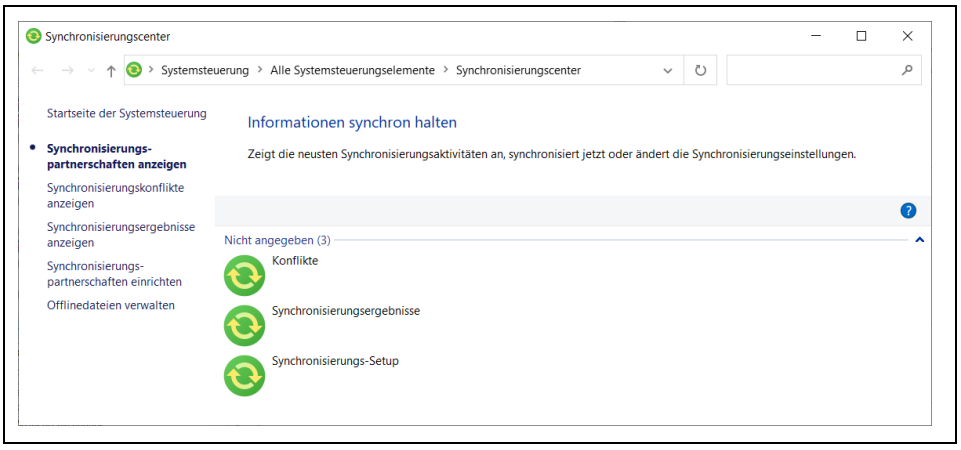

Bild 8. Synchronisationscenter

8 Start der Dateisignatur-Verifizierung

Das Programm *SigVerif* sucht nach Dateien, die nicht miz einer digitalen Signatur versehen sind.

Codeliste 6. Die Testprozedur in ThisDocument startet die Dateisignatur-Verifizierung

```
Sub TestShellSigVerif()
   Dim vReturn As Variant
   vReturn = Shell("SIGVERIF.EXE", 1)
End Sub
```

Mit dem Start öffnet sich das Dialogfenster zur Verifizierug (Bild 9).

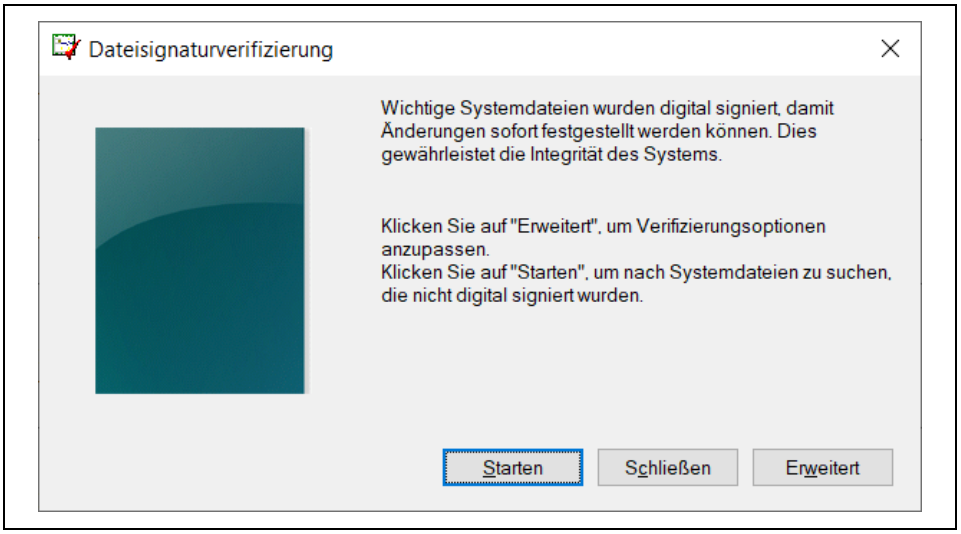

Bild 9. Dialogfenster zur Dateisignatur-Verifizierung

9 Start der Treiberprüfung

Das Programm *Verifier* startet den Manager zur Treiberprüfung. Damit können alle Einstellungen zu Treiber-Zertifikaten festgelegt werden.

Codeliste 7. Die Testprozedur in ThisDocument startet den Treiber-Manager

```
Sub TestShellVerifier()
   Dim vReturn As Variant
   vReturn = Shell("VERIFIER.EXE", 1)
End Sub
```

Mit dem Start öffnet sich das Dialogfenster zur Treiberprüfung (Bild 10).

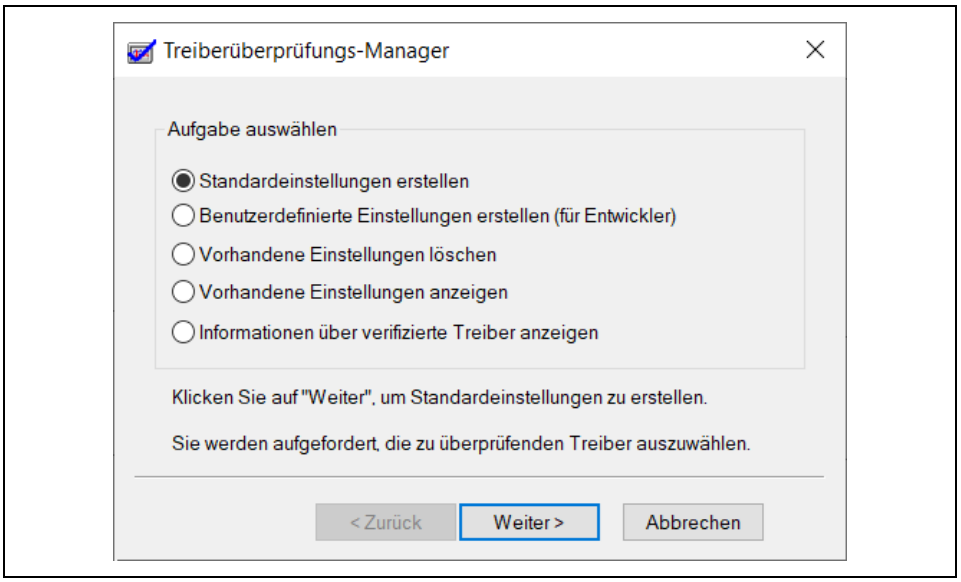

Bild 10. Treiberüberprüfungs-Manager

10 Start der Windows-Script-Host-Verwaltung

Das Programm *WScript* startet die Verwaltung der Script-Host-Einstellungen.

Codeliste 8. Die Testprozedur in ThisDocument startet die Windows-Script-Verwaltung

```
Sub TestShellWScript()
   Dim vReturn As Variant
   vReturn = Shell("WSCRIPT.EXE", 1)
End Sub
```

Mit dem Start öffnet sich das Dialogfenster-Windows-Script-Host-Einstellungen (Bild 11).

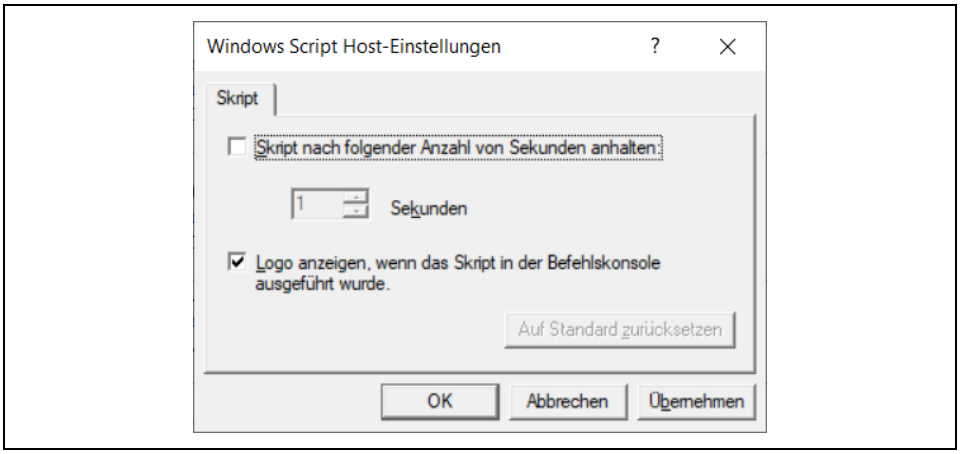

Bild 11. Dialogfenster zu Windows-Script-Host-Einstellungen