Documentation by Team Hahner® – Engineers of (Word) Solutions
Licensed version for visitors to the Team Hahner® website

Word (Bug): Opening files in “foreign” formats from Word via VBA calls

This document is accessible in accordance with the requirements of the BFSG, BITV 2.0, and WCAG 2.1.

Version **1.01**

Author **Dipl.-Ing. (FH) Markus Hahner**
Team Hahner® – Engineers of (Word) Solutions
www.hahner.de | www.schauen-statt-lesen.de |
www.office-sicherheit.de

This documentation contains confidential and/or legally protected information and is intended solely for the recipient or group of recipients identified on the title page and in the footers. Unauthorized copying, partial extraction of content, and unauthorized disclosure of this documentation or excerpts from it in any form are prohibited. The content of this documentation was created by the author without the use of AI. Copilot was used for translation only.

In this documentation, personal designations are not stated in feminine, masculine, and non-binary forms. The generic masculine form used here generally includes all persons regardless of gender.

Hahner®
Engineers of Word Solutions

Table of Contents

1. Opening files in "foreign" formats from Word via VBA 4
2. Alternative to the ShellExecute API 8
3. Behavior of Other API Declarations 9
4. Enhancements/Versions 10
4.1 Planned Enhancements to This Documentation 10
4.2 Versionsverlauf der Dokumentation 10
List of Figures 11
About the Author/Trainer | Contact 12
Legal Notice 13

All screenshots, commands, and settings in this documentation are based on Windows 11/10 and the Microsoft 365 suite, but essentially also apply to current Office versions such as Office 2024.

For Office 365, the current version in the "Current Channel" update channel serves as the basis. The features released there are also incorporated into other Office 365 update channels, such as the "Monthly Enterprise Channel," after a delay. Compared with the retail version, the LTSC versions of Office 2024 and Office 2021 receive few feature updates but regular security updates; like Office 365, they use the Click-to-Run installation method.

The same also applies to the volume-license version of Office 2019, which is no longer supported by Microsoft. For the likewise unsupported Office 2016, a distinction is made between the retail and volume-license versions depending on the feature. Compared with the retail version, the volume-license version uses a different installation method (MSI instead of Click-to-Run) and receives fewer feature updates, but it also receives regular security updates.

Accordingly, Office 2021, 2019, 2016, 2013, 2010, and 2007 are now covered only in part by this documentation.

Microsoft support lifecycle (see https://docs.microsoft.com/de-de/lifecycle/):

- Extended support for Office 2007 ended on October 10, 2017.
- Extended support for Office 2010 ended on October 13, 2020.
- Extended support for Office 2013 ended on April 11, 2023.
- Extended support for Office 2016 and Office 2019 ended on October 14, 2025.
- Extended support for Office 2021 ended on October 12, 2026 (retail and LTSC versions).
- Extended support for Office 2024 ends on October 9, 2029 (retail and LTSC versions).

- Extended support for Windows 10 ended on October 14, 2025, for the latest release, 22H2 (Enterprise, Pro, Home, and Education). Under Microsoft's terms, updates will be provided for one additional year. With the appropriate registration, which requires a Microsoft account, support can be extended free of charge until October 12, 2027.
- Extended support for Windows 10 Enterprise LTSC 2021 ends on January 12, 2027.
- The final end-of-support date for Windows 11 has not yet been determined. Depending on the installed release, support end dates have already been defined for individual releases (see https://learn.microsoft.com/de-de/lifecycle/products/windows-11-home-and-pro).

All path information is based on the current Windows 11 Pro version as of the publication date of this document (20.10.2026).

Hahner



1. Opening files in "foreign" formats from Word via VBA

Open a PDF file, for example, from a VBA-based add-in? Until now, this was easily possible using the `ShellExecute` API function – in both the 32-bit and 64-bit versions of Word. To invoke it, only the following API declaration was required:

```
Public Declare PtrSafe Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" ( _
               ByVal hwnd As LongPtr, _
               ByVal lpOperation As String, _
               ByVal lpFile As String, _
               ByVal lpParameters As String, _
               ByVal lpDirectory As String, _
               ByVal nShowCmd As LongPtr) As Long
```

Thanks to the `PtrSafe` attribute, the API declaration works in both the 32-bit and 64-bit versions, so conditional compilation such as the following is not required to distinguish between the bit versions:

```
#If Win64 Then
    <64-Bit-Deklaration>
#Else
    <32-Bit-Deklaration>
#End If
```

For example, the API is called using the following routine, which receives the path (`svFilepath`) and the file name (`svFilename`) as parameters:

```
Sub pLoadExternalFile(ByVal svFilepath As String, ByVal svFilename As String)
    Dim lReturn as Long

    On Error Resume Next
    lReturn = ShellExecute(hwnd:=0, _
                           lpOperation:="Open", _
                           lpFile:=svFilename, _
                           lpParameters:="", _
                           lpDirectory:=svFilepath, _
                           nShowCmd:=3)
    On Error GoTo 0
End Sub
```

If necessary, the return value `lReturn` can be used to check whether the API call was successful. A value less than or equal to 32 indicates an error; a value greater than 32 indicates successful execution.

Since early to mid-October, depending on the Microsoft 365 update channel, the API has returned `0` and therefore no longer opens the specified file.

When the `pLoadExternalFile` routine is called within the VBA code – as shown in the following `pTest` routine – nothing happens. The PDF file is not opened.

```
Sub pTest()
    Dim sPath as String
    Dim sFilename as String

    sPath = "D:\Daten"
    sFilename = "Markus Hahner Testdatei.pdf"

    pLoadExternalFile svFilepath:=sPath, svFilename:=sFilename
End Sub
```

Other API declarations – such as those listed in Chapter 3, "Behavior of Other API Declarations" – continue to work correctly.

> **Note:** The behavior documented in this chapter is observed only in Word 365 and the retail version of Word 2024. The **`ShellExecute`** API continues to work correctly in Excel 365 and PowerPoint 365 with the calls described in this chapter. The behavior has also not been observed in Word 2024 LTSC.

The following overview shows the account information for the affected and unaffected versions:

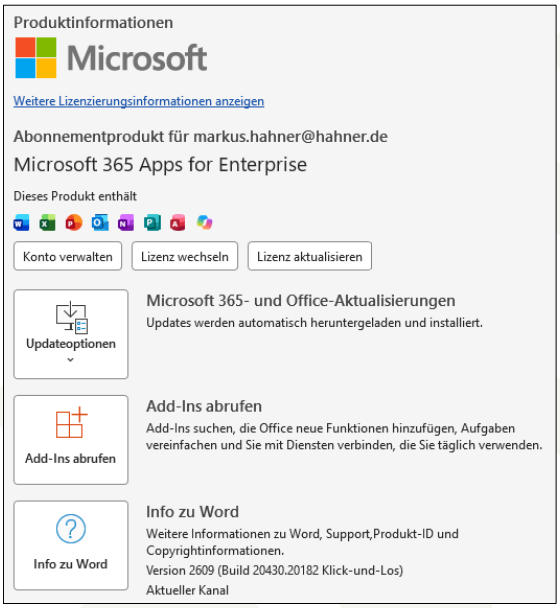


Figure 1: Account information for Microsoft 365 Apps for Enterprise, Current Channel
Currently **affected** version

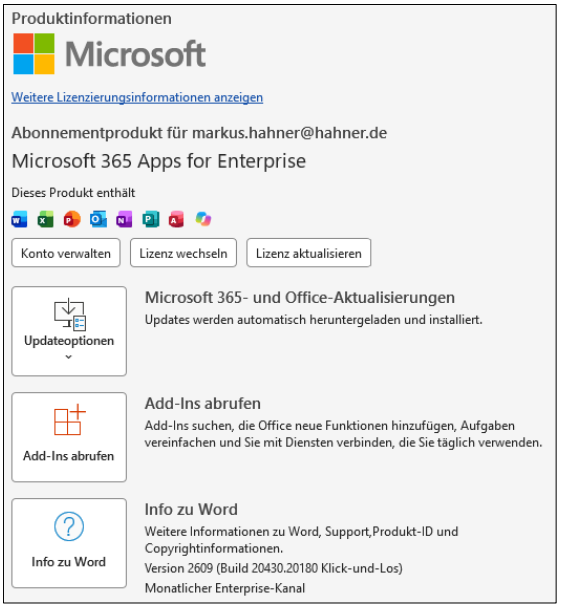


Figure 2: Account information for Microsoft 365 Apps for Enterprise, Monthly Enterprise Channel
Currently **affected** version

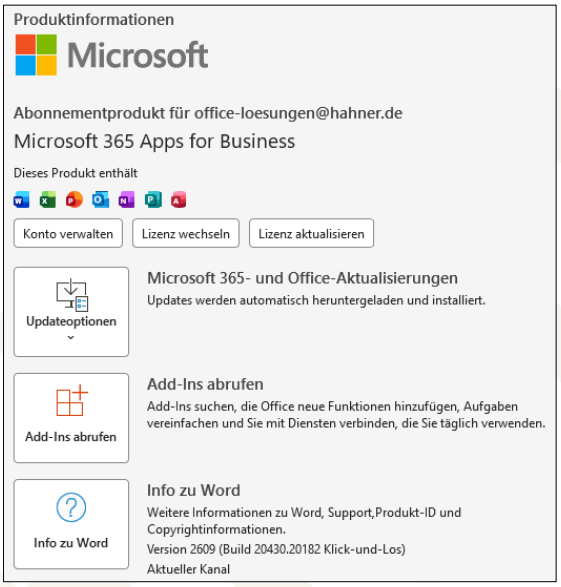


Figure 3: Account information for Microsoft 365 Apps for Business, Current Channel
Currently **affected** version



Figure 4: Account information for Microsoft Office Home & Business 2024, Current Channel
Currently **affected** version



Figure 5: Account information for Microsoft Office LTSC Professional Plus 2024
Currently **unaffected** version

2. Alternative to the ShellExecute API

To continue opening files in foreign formats through VBA, the **`WScript.Shell`** object can be used as a replacement for the **`ShellExecute`** API. The object is a Windows Script Host component ("WSH") and can be declared and called directly in VBA.

The following procedure replaces the procedure of the same name from Chapter 1 „Opening files in "foreign" formats from Word via VBA". It receives both the path (**`svFilepath`**) and the file name (**`svFilename`**), which are combined into one string within the procedure while accounting for a possible trailing backslash in the path.

```
Sub pLoadExternalFile(ByVal svFilepath As String, ByVal svFilename As String)
    Dim oWScriptShell As Object

    On Error Resume Next
    If Right(svFilepath, 1) <> "\" Then svFilepath = svFilepath & "\"
    Set oWScriptShell = CreateObject("WScript.Shell")
    oWScriptShell.Run Chr(34) & svFilepath & svFilename & Chr(34)
    Set oWScriptShell = Nothing
    On Error GoTo 0
End Sub
```

Because a space in the path or file name would cause the call to fail, the complete path is enclosed in double quotation marks (**`Chr(34)`**); this is not necessary when calling the **`ShellExecute`** API.

Hahner

3. Behavior of Other API Declarations

In testing, the limitation (bug?) documented in Chapter 1 „Opening files in "foreign" formats from Word via VBA" was observed only with the **`ShellExecute`** API. Other API calls, such as those below for reading the registry, continue to work correctly with the Word 365/2024 updates released in early/mid-October.

```
Public Declare PtrSafe Function RegOpenKey Lib "advapi32.dll" Alias "RegOpenKeyA" ( _
                ByVal hKey As LongPtr, _
                ByVal lpSubKey As String, _
                phkResult As Long) As LongPtr

Public Declare PtrSafe Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As LongPtr)
As LongPtr

Public Declare PtrSafe Function RegEnumValue Lib "advapi32.dll" Alias "RegEnumValueA" (
_
                ByVal hKey As LongPtr, _
                ByVal dwIndex As Long, _
                ByVal lpValueName As String, _
                lpcbValueName As Long, _
                ByVal lpReserved As Long, _
                lpType As Long, _
                lpData As Any, _
                lpcbData As Long) As LongPtr

Public Declare PtrSafe Function GetPrivateProfileString Lib "kernel32.dll" Alias
"GetPrivateProfileStringA" ( _
                ByVal lpApplicationName As String, _
                ByVal lpKeyName As Any, _
                ByVal lpDefault As String, _
                ByVal lpReturnedString As String, _
                ByVal nSize As LongPtr, _
                ByVal lpFileName As String) As LongPtr

Public Declare PtrSafe Function WritePrivateProfileString Lib "kernel32.dll" Alias
"WritePrivateProfileStringA" ( _
                ByVal lpApplicationName As String, _
                ByVal lpKeyName As Any, _
                ByVal lpString As Any, _
                ByVal lpFileName As String) As Long
```

4. Enhancements/Versions

Planned enhancements and changes made to this documentation are listed below.

4.1 Planned Enhancements to This Documentation

The following enhancements to this documentation are already planned or in progress:

- -/-

4.2 Versionsverlauf der Dokumentation

Version number	Publication date	Version information
v1.00	18.10.2026	■ Initial version
v1.01	20.10.2026	■ Added the `Set oWScriptShell = Nothing` statement to the `pLoadExternalFile` procedure in Chapter 2 „Alternative to the ShellExecute API" ■ Additional English-language version

Hahner

List of Figures

Figure 1: Account information for Microsoft 365 Apps for Enterprise, Current Channel Currently **affected** version 5
Figure 2: Account information for Microsoft 365 Apps for Enterprise, Monthly Enterprise Channel Currently **affected** version 6
Figure 3: Account information for Microsoft 365 Apps for Business, Current Channel Currently **affected** version 6
Figure 4: Account information for Microsoft Office Home & Business 2024, Current Channel Currently **affected** version 7
Figure 5: Account information for Microsoft Office LTSC Professional Plus 2024 Currently **unaffected** version 7

Hahner



About the Author/Trainer | Contact



Markus Hahner

Graduate Engineer (Dipl.-Ing. (FH))

+49 7720 810046

markus.hahner@hahner.de

www.hahner.de | www.schauen-statt-lesen.de | www.office-sicherheit.de

- Project-experienced engineer supporting SMEs, large enterprises, public authorities, and associations in the use of Microsoft 365 and Office rollouts
- Developer of template solutions and templates with tens of thousands of installations (see https://www.hahner.de/word-solutions/)
- Certified trainer (Microsoft Certified Educator, Microsoft MIE, Microsoft Certified Trainer from 2012 to 2026) specializing in Word, .NET, VBA, XML, Office security, and Microsoft 365 Apps (certifications: https://www.hahner.de/zertifizierungen/)
- Technical author of more than 50 books published worldwide, including titles from Microsoft Press; LinkedIn Learning/video2brain trainer for Word (see the German National Library catalog); technical editor of numerous IT books
- Video blogger (www.schauen-statt-lesen.de) und Office blogger (www.hahner.de)



Legal Notice

© 2015 – 2026, Team Hahner® – Engineers of (Word) Solutions, Dipl.-Ing. (FH) Markus Hahner

All data is protected by copyright. Copying is prohibited. All copyrights for data are reserved. Unauthorized downloading or other kinds of copying prohibited. No distribution or publication in printed or electronic form without the express written permission of Team Hahner® – Engineers of (Word) Solutions, Dipl.-Ing. (FH) Markus Hahner.

Team Hahner® – Engineers of (Word) Solutions
Dipl.-Ing. (FH) Markus Hahner, MCE/MCP/MIE/MOSM
Arndtstraße 28
78054 Villingen-Schwenningen
Germany

+49 7720 810046
Chat with Markus Hahner via Teams
info@hahner.de
www.hahner.de | www.schauen-statt-lesen.de | www.office-sicherheit.de

All screenshots in this documentation were created with Snagit®
(for details, see www.techsmith.de/snagit.html).

CAMTASIA, SNAGIT, AND TECHSMITH ARE TRADEMARKS OF TECHSMITH CORPORATION AND ARE REGISTERED AS SUCH IN THE UNITED STATES AND OTHER COUNTRIES.

Customized training courses, workshops & online seminars
on Snagit, Camtasia und Audacity | www.screencast-training.de

Version: 1.01 vom 10/20/2026