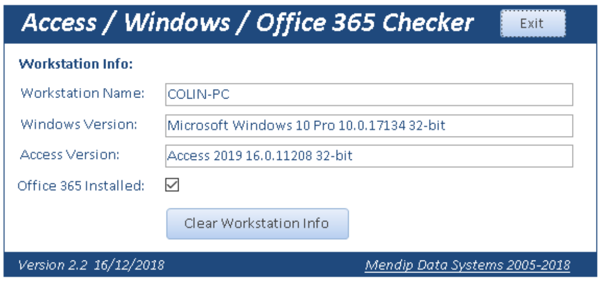
Using the Access Version Checker – Version 2.2

1. Using the application

When the application first opens, it will collect information about the version of Windows and access being used. It will also determine whether these are 32-bit or 64-bit and whether a copy of Office 365 is installed.

This process will take a few seconds and the result will look similar to one of the screenshots below:



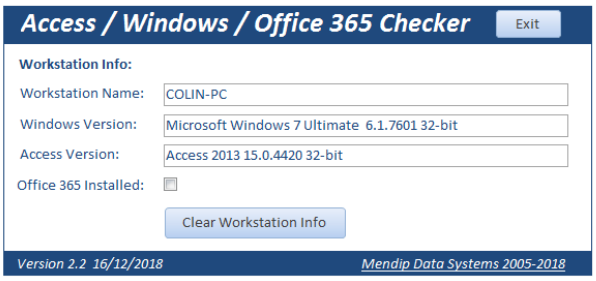
Access / Windows / Office 365 Checker

Workstation Info:

Workstation Name: COLIN-PC
Windows Version: Microsoft Windows 10 Pro 10.0.17134 32-bit
Access Version: Access 2019 16.0.11208 32-bit
Office 365 Installed: ☑

Version 2.2 16/12/2018 *Mendip Data Systems 2005-2018*

Access 2019 (365)



Access / Windows / Office 365 Checker

Workstation Info:

Workstation Name: COLIN-PC
Windows Version: Microsoft Windows 7 Ultimate 6.1.7601 32-bit
Access Version: Access 2013 15.0.4420 32-bit
Office 365 Installed: ☐

Version 2.2 16/12/2018 *Mendip Data Systems 2005-2018*

Access 2013 Retail

In the two examples shown above, a 32-bit version of Access is being run on 32-bit Windows
The left screenshot is for Access 365 and that on the right is for a retail version of Access 2013

2. How the Application Works

a) Various functions (in the module ***modSysInfo***) are used to identify the Windows & Access versions/bitnesses:
   - Windows – ***GetOperatingSystem / IsWin32OrWin64***
   - Access – ***GetAccessVersion / GetAccessEXEVersion / IsOfficex64***

   The results are stored in the table **tblComputerInfo**

   If the application is reopened on a new workstation or using a different version of Access, this data is deleted automatically and the table then repopulated.

b) Checking whether **Office 365** is installed is more difficult.
   As Microsoft uses the same version numbers for both retail Office and the Office 365 subscription model, the approach used is to check the registry.
   The code to do this is in the ***CheckAccess365*** function (in ***modFunctions***).

   Versions of Access prior to 2007 include this registry key:
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\JET\4.0\Engines

   All retail versions of Access from 2007 onwards include a registry key similar to:
   *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\**16.0**\Access Connectivity Engine\Engines*
   The part in **bold** is the Access version number

   For 32-bit Access on 64-bit Windows, this changes to:
   *HKEY_LOCAL_MACHINE\SOFTWARE\**Wow6432Node**\Microsoft\Office\16.0\Access Connectivity Engine\Engines*

   In each case, the Engines key contains a string ***SystemDB*** with value ***system.mdb***

   NOTE:
   See section 3 below for more information about the **Wow6432Node** section of the registry

However, for Office 365 installations, a **ClickToRun** registry structure is used instead.

For 32-bit Windows, only 32-bit Access can be installed.
The function will check for the existence of the registry key:
*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\**ClickToRun\REGISTRY\MACHINE**\Software\Microsoft\Office\16.0\Access Connectivity Engine\Engines*

For 64-bit Access in 64-bit Windows, the same registry key is used as above

However, for 32-bit Access 365 on 64-bit Windows, this changes to:
*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\**ClickToRun\REGISTRY\MACHINE\Software\Wow6432Node\Microsoft\Office\**16.0\Access Connectivity Engine\Engines*

So, if one of the above keys is found, Office 365 is installed.

However, it is not that ‘simple’. If a **retail version** of **Office 2013/2016/2019** is installed but the user enters their **Microsoft account** information either during installation or at a later time, this triggers the **ClickToRun** registry structure to be created!

In other words, it is then treated as Office 365 even though it is still a retail product.
However, the software is not updated with new features as is the case with a true Office 365 product

Furthermore, in such cases, a slightly different registry structure MAY also be created such as:
*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\**15.0**\ClickToRun\REGISTRY\MACHINE\Software\Microsoft\Office\15.0\Access Connectivity Engine\Engines*
or
*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\**16.0**\ClickToRun\REGISTRY\MACHINE\Software\Wow6432Node\Microsoft\Office\16.0\Access Connectivity Engine\Engines*

As a result, the code looks for any of those key structures to determine whether Office 365 is installed.
Why Microsoft decided to make this so very difficult is hard to understand!

c) Once the correct registry key for the **SystemDB** string value has been determined, this is also stored in the table **tblComputerInfo**.

3. *Reading the registry Wow6432Node*

When using 32-bit applications in 64-bit Windows, registry entries are made to the **Wow6432Node.**
Windows uses a process called **registry redirection** to manage this area but that causes issues when trying to view those entries from a 32-bit application such as Access
This area cannot be read from or written to using standard code as used on 'pure' 32-bit or 64-bit systems.
Instead, I have used 'special' code which is elevated to 64-bit as and when required

```
Public Function GetStdRegProv() As Object
' http://msdn.microsoft.com/en-us/library/aa394600(VS.85).aspx
'code updated by Jeff Holm to manage mixed mode systems

On Error GoTo ErrHandler:

   Dim strComputer As String

   strComputer = "."

   If IsWin32OrWin64 = IsOfficex64 Then    'Office & Windows bitness match, no need to force 64-bit

      Set GetStdRegProv = GetObject("winmgmts:" _
                     & "{impersonationLevel=impersonate}!\\" _
                     & strComputer & "\root\default:StdRegProv")

   Else    'Office 32-bit & Windows 64-bit, so have to elevate GetStdRegProv to 64-bit

      Dim objCtx As Object
      Dim objLocator As Object
      Dim objServices As Object
      Dim objStdRegProv As Object

      Set objCtx = CreateObject("WbemScripting.SWbemNamedValueSet")
      objCtx.Add "__ProviderArchitecture", 64
      objCtx.Add "__RequiredArchitecture", True
      Set objLocator = CreateObject("Wbemscripting.SWbemLocator")
      Set objServices = objLocator.ConnectServer(strComputer, "root\default", "", "", , , , objCtx)
      Set GetStdRegProv = objServices.Get("StdRegProv")

      Set objServices = Nothing
      Set objLocator = Nothing
      Set objCtx = Nothing
   End If

Exit_ErrHandler:
   On Error Resume Next
   Exit Function

ErrHandler:
   If Err.Number >= 0 Then
      MsgBox "Error " & Err.Number & " in GetStdRegProv procedure: " & Err.description, vbOKOnly +
vbCritical
   End If
   Resume Exit_ErrHandler

End Function
```

4. Acknowledgments

I am extremely grateful to Utter Access forum member **Jeff Holm** for repeatedly testing different versions of this application in mixed **32/64 bit** systems. Also for making several valuable suggestions and providing code snippets used for solving issues with registry keys using the **Wow6432Node** without having to deal with the complexities of ***registry redirection***.

Also, thanks are due to Tom Stiphout, Dev Ashish and Daniel Pineault for various items of standard code used in this application'

I would be grateful for any feedback related to this application.
To do so, please email me at: info@mendipdatasystems.co.uk.
Alternatively, use the contact page on my website: http://www.mendipdatasystems.co.uk

Colin Riddington *Mendip Data Systems*