OpenAxiom Windows Installer Script

Alfredo Portes

July 14, 2013

Abstract

This document contains the source code of the script to create an OpenAxiom installer for the Windows operating system. This script is based on the *Nullsoft Scriptable Install System* (NSIS)[1]. NSIS is a professional open source system to create Windows installers.

Contents

1 Introduction

This script is based on the NSIS program. This one can be downloaded from: http://nsis.sourceforge.net/Download. There is extensive documentation about its use and the commands to create and modify these installation scripts. Many examples are provided with the intallation files of NSIS.

To extract the NSIS script from this document, type:

```
notangle openaxiom.pamphlet >openaxiom.nsi
```

where `OpenAxiom.nsi` is the script to be compiled with NSIS. To extract the latex documentation type:

```
noweave -delay OpenAxiomWin.pamphlet >OpenAxiomWin.tex
latex OpenAxiomWin.tex or pdflatex OpenAxiomWin.tex
```

⟨*Header*⟩≡

```
  #OpenAxiom NSI Install Script
  #Written By: Dan Martens dan_martens@lycos.com
  #Updated By: Bill Page  bill.page1@sympatico.ca
  #Updated By: Alfredo Portes alfredo.portes@gmail.com
```

2 Header Files

A few header files are needed to provide some functionalities to the script.

- MUI.nsh provides the Graphical Interface for the script. It stands for Modern User Interface.
- StrFunc.nsh
- WinMessages.nsh

⟨*Header Files*⟩≡

```
    !include "MUI.nsh"
    !include "StrFunc.nsh"
    !include "WinMessages.nsh"
```

3 Variable Declarations

⟨*Variable Declarations*⟩≡
```
Var OpenAxiom_TEMP
Var STARTMENU_FOLDER

!verbose 3
!ifdef ALL_USERS
  !define WriteEnvStr_RegKey \
   'HKLM "SYSTEM\CurrentControlSet\Control\Session Manager\Environment"'
!else
  !define WriteEnvStr_RegKey 'HKCU "Environment"'
!endif
!verbose 4
```

- `APPNAME`: Consists of the name we want to give to the application.
- `BUILD_VERSION`: This is an identifier variable. It can consist of a version name we would like to append to the name of the application. Eg. OpenAxiom-1.0.
- `APPNAMEANDVERSION`: It is just a concatenation of the previous two variables and it will be the final name for the installer.

⟨*Variable Declarations*⟩+≡
```
; Define your application name
!define APPNAME "OpenAxiom"
!define BUILD_VERSION "1.1.0"
!define APPNAMEANDVERSION "${APPNAME}-${BUILD_VERSION}"
```

4 Main Script Section

This section contains the code and documentation of the core of the installation script. This code uses most of the functions described in the previous section.

When installing, we define the type of installation this will be. In this case this is more of a fancy display more than any practical use, given ”Typical” can be also customized.

⟨*Main Script*⟩≡
```
InstType "Typical"
```

The `Name` command creates a name for the application from the `APPNAMEANDVERSION` variable. The `OutFile` command creates the name of the executable file for the installer.

⟨*Main Script*⟩+≡
```
Name "${APPNAMEANDVERSION}"
OutFile "OpenAxiom-${BUILD_VERSION}.exe"
```

The `InstallDir` command defines where the files will be installed. `$PROGRAMFILES` is a NSIS glbal variable that points to the program files directory in Windows. Together with the variable `APPNAME` this defines the final location of the application files.

⟨*Main Script*⟩+≡
```
;Default installation folder
InstallDir "$PROGRAMFILES\${APPNAME}"
```

The name of the application needs to be saved in the Windows registry.

⟨*Main Script*⟩+≡
```
;Get installation folder from registry if available
InstallDirRegKey HKLM "Software\${APPNAME}" ""
```

The next command is apparently a fix for Windows Vista.

⟨*Main Script*⟩+≡
```
;Vista redirects $SMPROGRAMS to all users without this
RequestExecutionLevel admin
```

4.1 GUI Modifiable Screens

This section describes the creation and how to modify the various screens presented to the user during installation. Some of these screens are kept with the default values. However these can be changed to provide a more customized look and feel.

⟨*GUI Screens*⟩≡
```
;!define OpenAxiom_ABORTWARNING
!define MUI_ABORTWARNING
```

When the installation is complete, we present the user the option to run OpenAxiom immediately. To do this, we need to provide what is going to be the final location of the `open-axiom` executable.

⟨*GUI Screens*⟩+≡
```
;!define MUI_FINISHPAGE_RUN "$INSTDIR\bin\open-axiom.exe" '--system="$INSTDIR\lib\op
```

Add a little reminder and link for the user in the last page of the installer to donate to the Axiom Foundation.

⟨*GUI Screens*⟩+≡
```
!define MUI_FINISHPAGE_LINK "Please donate to the Axiom Foundation"
!define MUI_FINISHPAGE_LINK_LOCATION "http://axiom-developer.org/public/donate.html"
```

Generate a generic "Welcome" window. This window will have a basic greeting, providing the name of the application.

⟨*GUI Screens*⟩+≡
```
!insertmacro MUI_PAGE_WELCOME
```

Provide the License for OpenAxiom as a window showing a License agreement to the user. We need to provide a location to the file.

⟨*GUI Screens*⟩+≡
```
!insertmacro MUI_PAGE_LICENSE "OpenAxiom\License.txt"
```

Place the proper shorcuts for OpenAxiom in the startmenu and place the appropiate entries in the Windows registry.

⟨*GUI Screens*⟩+≡
```
!insertmacro MUI_PAGE_COMPONENTS
!insertmacro MUI_PAGE_DIRECTORY

;Start Menu Folder Page Configuration
!define MUI_STARTMENUPAGE_REGISTRY_ROOT "HKCU"
!define MUI_STARTMENUPAGE_REGISTRY_KEY "Software\${APPNAME}"
!define MUI_STARTMENUPAGE_REGISTRY_VALUENAME "Start Menu Folder"

!insertmacro MUI_PAGE_STARTMENU Application $STARTMENU_FOLDER
!insertmacro MUI_PAGE_INSTFILES
!insertmacro MUI_PAGE_FINISH
```

The following screens are the ones to be displayed to the user during installation.

⟨*GUI Screens*⟩+≡
```
!insertmacro MUI_UNPAGE_WELCOME
!insertmacro MUI_UNPAGE_CONFIRM
!insertmacro MUI_UNPAGE_INSTFILES
!insertmacro MUI_UNPAGE_FINISH
```

4.2 Language Declaration

Various possible languages are provided for the installer. If more languages support is needed, they should be add here. The first language, in this case English, will be the default language.

⟨*Language Declarations*⟩≡
```
  !insertmacro MUI_LANGUAGE "English"
  !insertmacro MUI_LANGUAGE "French"
  !insertmacro MUI_LANGUAGE "German"
  !insertmacro MUI_LANGUAGE "Russian"
  !insertmacro MUI_LANGUAGE "Spanish"
  !insertmacro MUI_LANGUAGE "TradChinese"
  !insertmacro MUI_RESERVEFILE_LANGDLL
```

4.3 OpenAxiom Core Installation

This section describes the copying of the OpenAxiom files to the destination directory. The files to be copied need to be placed in a directory called `OpenAxiom`. This can be changed by modifying the line:

```
File /r OpenAxiom\*.*
```

The installer will copy all the files contained here recursevely to the location specified in the `$INSTDIR` variable.

⟨*OpenAxiom Core Section*⟩≡
```
  Section "!OpenAxiom Core" Section1

    SectionIn 1 2 RO

    ; Set Section properties
    SetOverwrite on

    SetOutPath "$INSTDIR"

    File /r OpenAxiom\*.*

    ReadEnvStr $0 "USERPROFILE" ;

    ;Store installation folder
    WriteRegStr HKCU "Software\OpenAxiom" "" $INSTDIR

    ;Create uninstaller
    WriteUninstaller "$INSTDIR\Uninstall.exe"
```

We specify the different shortcuts we want in the start menu and in the desktop, like the `open-axiom` executable and other shorcuts.

⟨*OpenAxiom Core Section*⟩+≡
```
    !insertmacro MUI_STARTMENU_WRITE_BEGIN Application

      CreateDirectory "$SMPROGRAMS\$STARTMENU_FOLDER"

      CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\${APPNAME}.lnk" "$INSTDIR\bin\open-a
      CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\Uninstall.lnk" "$INSTDIR\Uninstall.e
      CreateShortCut "$DESKTOP\OpenAxiom.lnk" "$INSTDIR\bin\open-axiom.exe" '--system="$
      CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\OpenAxiom Website.lnk" "http://www.o
      CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\OpenAxiom Bug Reports.lnk" "http://w
      CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\Donate to Axiom Foundation.lnk" "htt

    !insertmacro MUI_STARTMENU_WRITE_END

  SectionEnd
```

4.4 Documentation Installation

Like the previous section was about copying the core OpenAxiom files, this section describes the copying of the documentation files in the installation directory. The documentation files need to be placed in a directory called *doc*.

```
File /r doc
```

Here, we also add two shorcuts to the Axiom book and the Axiom tutorial pdf files. These two files need to be located also in the *doc* directory.

⟨*Documentation Section*⟩≡
```
  Section /o "Documentation" Section2

    SetOverwrite on
    SetOutPath "$INSTDIR"

    File /r doc

    ;Shortcuts
    CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\Axiom Tutorial.lnk" "$INSTDIR\doc\tuto
    CreateShortCut "$SMPROGRAMS\$STARTMENU_FOLDER\Axiom Book.lnk" "$INSTDIR\doc\axiom.pd

  SectionEnd
```

4.5 Source Code Installation

The OpenAxiom source code is placed in a directory called *src* and it will be copied to the installation directory.

⟨*Source Code Section*⟩≡
```
Section /o "Source Code" Section3

  ; Set Section properties
  SetOverwrite on

  ; Set Section Files and Shortcuts
  SetOutPath "$INSTDIR"

  File /r src

SectionEnd
```

4.6 Finish Core Installation

⟨*Finish Section*⟩≡
```
Section -FinishSection

  SetOutPath "$0\My Documents" # sets the 'START IN' parameter
  WriteRegStr HKLM "Software\${APPNAME}" "" "$INSTDIR"
  WriteRegStr HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\${APPNAME}" "Di
  WriteRegStr HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\${APPNAME}" "Ur
  WriteUninstaller "$INSTDIR\uninstall.exe"

SectionEnd
```

4.7 Modern install component descriptions

Probably this section should be moved somewhere else. These descriptions appear when the user is selecting what is going to be installed (Core, Documentation, Source). These are no more than tool-tips to describe these options.

⟨*Finish Section*⟩+≡
```
!insertmacro MUI_FUNCTION_DESCRIPTION_BEGIN
  !insertmacro MUI_DESCRIPTION_TEXT ${Section1} "The main program files."
  !insertmacro MUI_DESCRIPTION_TEXT ${Section2} "Program Documentation"
  !insertmacro MUI_DESCRIPTION_TEXT ${Section3} "Source code"
!insertmacro MUI_FUNCTION_DESCRIPTION_END
```

⟨*Finish Section*⟩+≡
```
Section -AddtoPath
  Push "$INSTDIR\bin"
  Call AddToPath
SectionEnd
```

4.8 Uninstaller Section

The purpose of the `Uninstall` section is to reverse everything done in the previous intallation sections. This means to delete the files and shortcuts created by the installation process. This section needs to be handle carefully because other shortcuts and files for other applications can be deleted.

First, delete the uninstall file and then recursevely the directory in which OpenAxiom is installed with all its contents.

⟨*Uninstaller Section*⟩≡
```
Section "Uninstall"

  Delete "$INSTDIR\Uninstall.exe"
  RMDir /r $INSTDIR
```

⟨*Uninstaller Section*⟩+≡
```
  !insertmacro MUI_STARTMENU_GETFOLDER Application $OpenAxiom_TEMP

  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\Uninstall.lnk"
  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\${APPNAME}.lnk"
  Delete "$DESKTOP\${APPNAME}.lnk"
  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\OpenAxiom Website.lnk"
  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\OpenAxiom Bug Reports.lnk"
  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\Donate to Axiom Foundation.lnk"
  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\Axiom Tutorial.lnk"
  Delete "$SMPROGRAMS\$OpenAxiom_TEMP\Axiom Book.lnk"

  ;Delete empty start menu parent directories
  StrCpy $OpenAxiom_TEMP "$SMPROGRAMS\$OpenAxiom_TEMP"

  startMenuDeleteLoop:
        ClearErrors
    RMDir $OpenAxiom_TEMP
    GetFullPathName $OpenAxiom_TEMP "$OpenAxiom_TEMP\.."

    IfErrors startMenuDeleteLoopDone

    StrCmp $OpenAxiom_TEMP $SMPROGRAMS startMenuDeleteLoopDone startMenuDeleteLoop
  startMenuDeleteLoopDone:

  DeleteRegKey /ifempty HKCU "Software\${APPNAME}"
  Push "$INSTDIR\bin"

  Call un.RemoveFromPath

SectionEnd
```

5 Function Declarations

The functions declared in this section are used to provide auxiliary functionality to this script. Things like adding/removing the OpenAxiom executable to/from the system PATH is done the by different functions described in this section. The vast majority of these functions cannot be currently properly documented. This is because they were taken from undocumented examples in the web to achieve certain functionalty. Hopefully people reading this document can contribute to their documentation.

5.1 Function to add OpenAxiom executable to the PATH

The `AddToPath` function adds the given value in `dir` to the search path. Its input is the head of the stack and the value of the `dir` varaible is added at the beginning of the system path. Win9x systems may require to reboot.

⟨*Function Add To Path*⟩≡
```
Function AddToPath
  Exch $0
  Push $1
  Push $2
  Push $3

  # don't add if the path doesn't exist
  IfFileExists $0 "" AddToPath_done

  ReadEnvStr $1 PATH
  Push "$1;"
  Push "$0;"
  Call StrStr
  Pop $2
  StrCmp $2 "" "" AddToPath_done
  Push "$1;"
  Push "$0\;"
  Call StrStr
  Pop $2
  StrCmp $2 "" "" AddToPath_done
  GetFullPathName /SHORT $3 $0
  Push "$1;"
  Push "$3;"
  Call StrStr
  Pop $2
  StrCmp $2 "" "" AddToPath_done
  Push "$1;"
  Push "$3\;"
  Call StrStr
  Pop $2
  StrCmp $2 "" "" AddToPath_done

  Call IsNT
  Pop $1
  StrCmp $1 1 AddToPath_NT
    ; Not on NT
    StrCpy $1 $WINDIR 2
    FileOpen $1 "$1\autoexec.bat" a
    FileSeek $1 -1 END
    FileReadByte $1 $2
    IntCmp $2 26 0 +2 +2 # DOS EOF
      FileSeek $1 -1 END # write over EOF
    FileWrite $1 "$\r$\nSET PATH=$3;%PATH%$\r$\n"
    FileClose $1
    SetRebootFlag true
    Goto AddToPath_done

  AddToPath_NT:
    ReadRegStr $1 HKCU "Environment" "PATH"
    StrCpy $2 $1 1 -1 # copy last char
    StrCmp $2 ";" 0 +2 # if last char == ;
      StrCpy $1 $1 -1 # remove last char
    StrCmp $1 "" AddToPath_NTdoIt
      StrCpy $0 "$0;$1"
    AddToPath_NTdoIt:
      WriteRegExpandStr HKCU "Environment" "PATH" $0
      SendMessage ${HWND_BROADCAST} ${WM_WININICHANGE} 0 "STR:Environment" /TIMEOUT=50

  AddToPath_done:
    Pop $3
    Pop $2
    Pop $1
    Pop $0
FunctionEnd
```

5.2 Function to remove the `OpenAxiom` executable from PATH

The function `RemoveFromPath` removes the reference to of the `OpenAxiom` executable from the path. Its input is the head of the stack.

⟨*Function Remove From Path*⟩≡

```
Function un.RemoveFromPath
  Exch $0
  Push $1
  Push $2
  Push $3
  Push $4
  Push $5
  Push $6

  IntFmt $6 "%c" 26 # DOS EOF

  Call un.IsNT
  Pop $1
  StrCmp $1 1 unRemoveFromPath_NT
    ; Not on NT
    StrCpy $1 $WINDIR 2
    FileOpen $1 "$1\autoexec.bat" r
    GetTempFileName $4
    FileOpen $2 $4 w
    GetFullPathName /SHORT $0 $0
    StrCpy $0 "SET PATH=%PATH%;$0"
    Goto unRemoveFromPath_dosLoop

    unRemoveFromPath_dosLoop:
      FileRead $1 $3
      StrCpy $5 $3 1 -1 # read last char
      StrCmp $5 $6 0 +2 # if DOS EOF
        StrCpy $3 $3 -1 # remove DOS EOF so we can compare
      StrCmp $3 "$0$\r$\n" unRemoveFromPath_dosLoopRemoveLine
      StrCmp $3 "$0$\n" unRemoveFromPath_dosLoopRemoveLine
      StrCmp $3 "$0" unRemoveFromPath_dosLoopRemoveLine
      StrCmp $3 "" unRemoveFromPath_dosLoopEnd
      FileWrite $2 $3
      Goto unRemoveFromPath_dosLoop
      unRemoveFromPath_dosLoopRemoveLine:
        SetRebootFlag true
        Goto unRemoveFromPath_dosLoop

    unRemoveFromPath_dosLoopEnd:
      FileClose $2
      FileClose $1
```

```
   StrCpy $1 $WINDIR 2
   Delete "$1\autoexec.bat"
   CopyFiles /SILENT $4 "$1\autoexec.bat"
   Delete $4
   Goto unRemoveFromPath_done

 unRemoveFromPath_NT:
  ReadRegStr $1 HKCU "Environment" "PATH"
  StrCpy $5 $1 1 -1 # copy last char
  StrCmp $5 ";" +2 # if last char != ;
   StrCpy $1 "$1;" # append ;
  Push $1
  Push "$0;"
  Call un.StrStr ; Find `$0;` in $1
  Pop $2 ; pos of our dir
  StrCmp $2 "" unRemoveFromPath_done
   ; else, it is in path
   # $0 - path to add
   # $1 - path var
   StrLen $3 "$0;"
   StrLen $4 $2
   StrCpy $5 $1 -$4 # $5 is now the part before the path to remove
   StrCpy $6 $2 "" $3 # $6 is now the part after the path to remove
   StrCpy $3 $5$6

   StrCpy $5 $3 1 -1 # copy last char
   StrCmp $5 ";" 0 +2 # if last char == ;
   StrCpy $3 $3 -1 # remove last char

   WriteRegExpandStr HKCU "Environment" "PATH" $3
   SendMessage ${HWND_BROADCAST} ${WM_WININICHANGE} 0 "STR:Environment" /TIMEOUT=50

 unRemoveFromPath_done:
  Pop $6
  Pop $5
  Pop $4
  Pop $3
  Pop $2
  Pop $1
  Pop $0
FunctionEnd
```

6 Utility Functions

6.1 Function IsNT

The `IsNT` do no take any input. Its output can be obtained from the top of the stack = 1 if NT or 0 if not. The following is an example of how to use the function:

```
Call IsNT
Pop $R0
```

$R0 at this point is 1 or 0.

⟨*Utility Functions*⟩≡
```
  !macro IsNT un
  Function ${un}IsNT
    Push $0
    ReadRegStr $0 HKLM "SOFTWARE\Microsoft\Windows NT\CurrentVersion" CurrentVersion
    StrCmp $0 "" 0 IsNT_yes
    ; we are not NT.
    Pop $0
    Push 0
    Return

    IsNT_yes:
      ; NT!!!
      Pop $0
      Push 1
  FunctionEnd
```

⟨*Utility Functions*⟩+≡
```
  !macroend
  !insertmacro IsNT ""
  !insertmacro IsNT "un."
```

6.2 Function StrStr

The `StrStr` function takes as input the top of stack, which contains the string to search for. The top of stack-1 is the string to search in. The output result is place at the top of stack (replaces with the portion of the string remaining) and it does not modify any other variables. The following is an example on how to use this function:

```
Push "this is a long ass string"
Push "ass"
Call StrStr
Pop $R0
```

The value of `$R0` at this point is "ass string".

⟨*Utility Functions*⟩+≡
```
  !macro StrStr un
  Function ${un}StrStr
  Exch $R1 ; st=haystack,old$R1, $R1=needle
    Exch    ; st=old$R1,haystack
    Exch $R2 ; st=old$R1,old$R2, $R2=haystack
    Push $R3
    Push $R4
    Push $R5
    StrLen $R3 $R1
    StrCpy $R4 0
    ; $R1=needle
    ; $R2=haystack
    ; $R3=len(needle)
    ; $R4=cnt
    ; $R5=tmp
    loop:
      StrCpy $R5 $R2 $R3 $R4
      StrCmp $R5 $R1 done
      StrCmp $R5 "" done
      IntOp $R4 $R4 + 1
      Goto loop
  done:
    StrCpy $R1 $R2 "" $R4
    Pop $R5
    Pop $R4
    Pop $R3
    Pop $R2
    Exch $R1
  FunctionEnd
  !macroend
  !insertmacro StrStr ""
```

```
!insertmacro StrStr "un."
```

7 Document Structure

⟨*⟩≡
  ⟨*Header*⟩
  ⟨*Header Files*⟩
  ⟨*Variable Declarations*⟩
  ⟨*Function Add To Path*⟩
  ⟨*Function Remove From Path*⟩
  ⟨*Utility Functions*⟩
  ⟨*Main Script*⟩
  ⟨*GUI Screens*⟩
  ⟨*Language Declarations*⟩
  ⟨*OpenAxiom Core Section*⟩
  ⟨*Documentation Section*⟩
  ⟨*Source Code Section*⟩
  ⟨*Finish Section*⟩
  ⟨*Uninstaller Section*⟩

References

[1] *Nullsoft Scriptable Install System Homepage.* http://nsis.sourceforge.net/. Accessed on August 24, 2007.