The Ins and Outs of INI files

by Clark Anderson

Windows _.INI files can be very useful. They can “remember” how an individual customer last used your application. An _.INI file can extend the usefulness of one application to other customers. I have one report generator program that is tailored to more than a half dozen different product areas by adjustments to the _.INI file and adding specific queries and reports to the otherwise identical databases.

Parameters can be provided to the application in many places: 1) buried deep in control properties or code, 2) defined up front in form declarations or global declarations, 3) defined by the application _.INI file or 4) entered by the user at run time. I will show you how easy it is to add flexibility to your application with an _.INI file.

Most _.INI files are test files that are small enough to be edited with “Notepad”. They are organized in sections, each with a bracketed [Header], followed by Parameter Name=Parameter Value. Within an _.INI file, any text preceded by a semicolon “;” is ignored. This is useful for temporarily “commenting out” INI data or for just providing comments.

These are some global definitions I will use in my example. You can see that I use a naming convention. The gstr_ prefix indicates that it is a global string variable or constant.

```
Option Compare Database   'Use database order for string comparisons
Option Explicit

Global Const gstr_version = "1.1"
Global gstr_date_fmt As String
Global gstr_AppTitle As String

Global gstr_WinDir As String
Global gstr_INIFile As String

'MsgBox global constants
Global Const MB_OKCANCEL = 1   ' OK and Cancel buttons from VB CONSTANT.TXT
Global Const IDCANCEL = 2      ' Cancel button pressed from VB CONSTANT.TXT

Global Const DateTimeFormat_USA = "mm/dd/yyyy hh:mm:ss" ' USA
Global Const DateTimeFormat_UK = "dd/mm/yyyy hh:mm:ss"  ' UK
Global gstr_DateTimeFormat As String
```

I really had trouble with these API declarations until someone told me that the ACCESS documentation had an error and ALL variables must be declared “ByVal". There are separate API functions for retrieving string or integer values but only API functions for writing string values. That is OK because it is easy to use the CStr(IntegerValue) conversion function to build a string. These declarations must appear in your code as one continuous line. They are shown here using both variable type declaration methods (e.g.: nSize% and nSize as Integer). I prefer the “as Integer” style because it is more like other languages that I use.

```
' API functions

Declare Function GetWindowsDirectory Lib "Kernel" (ByVal lpBuffer As String, ByVal nSize
As Integer) As Integer

Declare Function GetProfileString Lib "Kernel" (ByVal lpAppName As String, ByVal
lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal
nSize As Integer) As Integer

Declare Function GetPrivateProfileInt% Lib "Kernel" (ByVal lpAppName$, ByVal lpKeyName$,
ByVal nDefault%, ByVal lpFileName$)

Declare Function GetPrivateProfileString% Lib "Kernel" (ByVal lpAppName$, ByVal
lpKeyName$, ByVal lpDefault$, ByVal lpReturnedString$, ByVal nSize%, ByVal lpFileName$)

Declare Function WritePrivateProfileString% Lib "Kernel" (ByVal lpAppName$, ByVal
lpKeyName$, ByVal lpString$, ByVal lpFileName$)
```

Now that we have declared the API functions, let us put them to use.

This is a sample subroutine to retrieve INI data that I invoke while loading the first form. I usually put the code in the main Global.BAS file. It specifies the name and location of the _.INI file and retrieves the most important information from it. ReadIni uses some utility functions that I will describe later. In another application I have also used the _.INI file to specify the Data and Report Subdirectories

```
Sub ReadIni ()
'           SAMPLE METHODS FOR NAMING INI FILE
'ACCESS:
    gstr_INIFile = CurDir$ & "\" & Application.CurrentObjectName & ".INI"
'VISUAL BASIC:
'    If Len(gstr_WinDir) = 0 Then gstr_WinDir = get_win_dir()
'    gstr_INIFile = gstr_WinDir & "\" & app.EXEName & ".INI"

'   Get location of files
  gstr_DatabasePath = GetINI_Param("Databases", "Location")

'   Now add on the database file name...
  gstr_DatabaseName = gstr_DatabasePath & "DATA\" & GetINI_Param("Databases", "Name")

'   Now add on the report file name...
 gstr_ReportName = gstr_DatabasePath & "REPORTS\" & GetINI_Param("Databases", "Report")

'   Get application title
  gstr_AppTitle = GetINI_Param("Application", "Title") & " v" & gstr_version
'                   OR
'  gstr_AppTitle = GetWriteINI_Param("Application", "Title", "Test INI") & " v" &
gstr_version

End Sub
```

This utility function, `get_win_dir, will` completely identify the “Windows” directory path used by the client computer. Sometimes it is not the C: drive and sometimes the directory is \WIN95. Please notice that these API functions use what is referred to as “zstrings”. They are terminated with a zero value. To be on the safe side, the length must be provided. Before invoking the function, some “maximum buffer space” is allocated. The API function returns the length of the buffer space that is actually used. These utility functions illustrate methods for working with the “zstrings”.

```
Function get_win_dir () As String
  Dim cBuffer As String
  Dim nReturned As Integer
    cBuffer = Space$(255) & Chr$(0)

    nReturned = GetWindowsDirectory(cBuffer$, Len(cBuffer$))
    get_win_dir = Left$(cBuffer$, nReturned%)

End Function
```

This GetINI_Param utility function can retrieve the specified parameter from its section in the _.INI file. I have provided the _.INI filename and location to these utility functions in the global variable, gstr_INIFile. The first argument, Header, refers to the section label in brackets e.g.: [Application] that is used to organize an _.INI file. The second argument, INI_Param, identifies the specific parameter in the section e.g.: Title=Test Application. If the parameter has more than one value they are separated by commas. This sample function does not deal with a list of values. The function returns a string e.g.: Test Application. In the GetPrivateProfileString API function the third argument specifies a default value in case none is found. I chose to use this feature to provide an error message and stop the application. The last GetPrivateProfileString argument is used to provide the name and location of the INI file.

```
Function GetINI_Param (Header As String, INI_Param As String) As Variant

  Dim cBuffer As String * 80
  Dim nReturned As Integer
```

```
    If Len(gstr_WinDir) = 0 Then gstr_WinDir = get_win_dir()
  ' Call API function to read INI file
  nReturned% = GetPrivateProfileString(Header$, INI_Param$, "Invalid", cBuffer$,
Len(cBuffer$), gstr_INIFile)

    If Left$(cBuffer, nReturned%) = "Invalid" Then
      MsgBox "Cannot find " & INI_Param & " under [" & Header & "] in INI file " &
gstr_INIFile, 16, "Critical Error"
      End 'Stop Application
    End If

  GetINI_Param = Left$(cBuffer$, nReturned%)

End Function
```

This GetWriteINI_Param utility function has the additional capability to update the _.INI file if necessary. This is very useful when you have added a new feature to your application on a client/server network. The additional argument, Default, is the string value that is to be written into the _.INI file if no value was found. A MsgBox prompt notifies and requests approval for the addition to the _.INI file. If there is any failure an error message is provided and the application is stopped.

The WritePrivateProfileString API function changes or creates the INI file specified by the fourth argument. The first argument identifies the section, the second argument, the parameter name, and the third argument specifies the value.

```
Function GetWriteINI_Param (Header As String, INI_Param As String, Default As String) As
Variant

  Dim cBuffer As String * 80
  Dim nReturned As Integer, answer As Integer

    If Len(gstr_WinDir) = 0 Then gstr_WinDir = get_win_dir()
  ' Call API function to read INI file
  nReturned% = GetPrivateProfileString(Header$, INI_Param$, "Invalid", cBuffer$,
Len(cBuffer$), gstr_INIFile)

    If Left$(cBuffer, nReturned%) = "Invalid" Then
       answer = MsgBox("Have not found " & INI_Param & " under [" & Header & "] in INI
file " & gstr_INIFile & Chr(13) & "will add value: " & Default, MB_OKCANCEL, "INI Update
Service")
       If answer = IDCANCEL Then Exit Function

  nReturned% = WritePrivateProfileString(Header$, INI_Param$, Default, gstr_INIFile)
  cBuffer = Space(80)
  nReturned% = GetPrivateProfileString(Header$, INI_Param$, "Invalid", cBuffer$,
Len(cBuffer$), gstr_INIFile)

    End If

    If Left$(cBuffer, nReturned%) = "Invalid" Then
      MsgBox "On 2nd try Cannot find " & INI_Param & " under [" & Header & "] in INI file
" & gstr_INIFile, 16, "Critical Error"
      End  'Stop
    End If

    GetWriteINI_Param = Left$(cBuffer$, nReturned%)

End Function
```

I have provided an example of why you might want to obtain information from the WIN.INI file. This get_date_format Utility function checks the WIN.INI file for the Short Date format to define a Global string variable that you can use in any

Date Time formats to Internationalize your application. So far I have only used it for the USA and the UK to control form display and input format. This may be useful in planning for the year 2000.

```
Function get_date_format () As String
    Dim cBuffer As String
    Dim nReturned As Integer
    Dim Short_Date As String
    Const Header = "intl"
    Const INI_Param = "sShortDate"

    cBuffer = Space$(255)

    nReturned% = GetProfileString(Header$, INI_Param$, "zzz", cBuffer$, Len(cBuffer$))

    Short_Date = Left$(cBuffer$, nReturned%)

    If InStr(Short_Date, "M/") < InStr(Short_Date, "d/") Then
      get_date_format = DateTimeFormat_USA
    Else
      get_date_format = DateTimeFormat_UK
    End If

End Function
```