    Functions For Working With The Registry

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | If you need to store information from one Excel session to the next, such as user preferences or application configuration data, you can store that data in the System Registry.  This page describes about 10 VBA functions the are used to read and write keys and values to the system Registry.  There are about an addition 10 functions that support the primary registry-related functions. For information about manually working with the Registry using the RegEdit program, see the [Modify The System Registry page](http://www.cpearson.com/excel/ModifyRegistry.htm).  See [this page](http://www.cpearson.com/excel/RegistryWorx.aspx) for a DLL component for working with the registry with VBA-friendly functions.  The procedures presented here should give you full control over the Registry. It is assumed that you are familiar with the system Registry. Note that there is no "undo" functionality when dealing with the System Registry and that if you delete or change a key or value, you may cause serious problems with Windows, up to and including not being able to start your system.  It is up to you to ensure that you are not deleting or changing critical system-related keys.  While the procedures described on this page will read, write, create and delete any registry key, you should use only your own registry keys. You can create keys for your own application, storing them in the HKEY\_CURRENT\_USER section, with a key named something similar to  "Software\Pearson\ImportMultiModules".  Overall, the Registry is organized much like the Windows file system. It is a hierarchical system, where keys may contain keys that contain values that have a specific value. The word "value" is used in two separate but related contexts. A key may contain one or more named "values", each of which contains either a String or Long value.  For example, the key "HKEY\_CURRENT\_USER\Software\Microsoft\Office\12.0\Excel\Options" contains many named values, each of which has a string or numeric value, such as the "AltStartup" value which has value equal to the folder you specified as an alternative startup directory for Excel. In the procedures described on this page and available in the downloadable files, the term "ValueName" is used to specify the named value (e.g, "AltStartup") and the term "ValueValue" is used to specify the contents of ValueName. For example, "ValueName" might refer to "AltStartup" and "ValueValue" refers to the contents of "AltStartup", such as "C:\XLStart".  The system registry is divide into parts or sections call *hives*. A *hive* is a distinct set of keys and their values. For example, the key "Software\Pearson\ImportMultiModules" and all of the values within this key comprise a hive. As long as you manipulate keys and value only within your own hive, you shouldn't encounter any problems.  You can examine the contents of the Registry and add, change, or delete keys and values using RegEdit program. On the Windows Start menu, choose Run, and enter RegEdit. This will start the Registry Editor program. Remember that all edits to the registry are done "live".  Once you change or delete a key or value, there is no way to undo that action or exit RegEdit without a save. Before you modify the system registry, be *sure* you are working with the proper key in the proper hive.  Note that there is nothing specific to Excel in the code. This code can be used in any application that supports VBA or in VB6.  These functions support reading and writing values of either String or Long data types. If you attempt to store another numeric data type (e.g., Double), it will be converted to a Long, and thus there is the possibility of data loss (the fractional portion of the number will be lost and the integer portion may be rounded). If you need to store a Single or Double value, convert it to a string using the CStr function and store it as a String data type. Incompatible types like objects, arrays, and user-defined types will cause an error to occur.  You can [download a bas module here](http://www.cpearson.com/Zips/modRegistry.zip) or a [complete workbook here](http://www.cpearson.com/Zips/Registry.zip). These functions require the [modGetSystemErrorMessageText](http://www.cpearson.com/Zips/modFormatMessage.zip) module that retrieves text descriptions of system error numbers. You can [read about this module here](http://www.cpearson.com/excel/_vti_cnf/FormatMessage.htm) or [download it here](http://www.cpearson.com/Zips/modFormatMessage.zip). You can download the [modRegistry bas module file here](http://www.cpearson.com/Zips/modRegistry.zip) or a [complete workbook here](http://www.cpearson.com/Zips/Registry.zip). The procedures in the module and described on this page call upon one another. You are strongly urged to import the entire module into your project rather than copy/pasting individual procedures.  **Error Reporting** Errors that may arise in the execution of the procedures are reported in 4 Public variables.  **Public G\_Reg\_AppErrNum As Long** This variable contains the error number assigned by the module procedures. This will be one of the public constants defined in the module beginning with C\_REG\_ERR\_.  **Public G\_Reg\_AppErrText As String** This is a text description of G\_Reg\_AppErrNum    **Public G\_Reg\_SysErrNum As Long** This is the system error number, the error number returned by the Registry API functions.  **Public G\_Reg\_SysErrText As String** This is a text description of G\_Reg\_SysErrNum.  If a function returns False (or Null) indicating that the operation was unsuccessful, you should examine the variable listed above to determine the cause of the error.  In all functions, BaseKey is one of the following:  HKEY\_CURRENT\_USER  HKEY\_LOCAL\_MACHINE  HKEY\_CLASSES\_ROOT  HKEY\_CURRENT\_CONFIG  HKEY\_DYN\_DATA  HKEY\_PERFORMANCE\_DATA  HKEY\_USERS  It is *strongly* recommended that you modify *only* keys in HKEY\_CURRENT\_USER.  The functions provided in the module are as follows.  **RegistryCreateKey**  Public Function RegistryCreateKey(BaseKey As Long, KeyName As String) As Boolean  This function  create a new key named KeyName in the BaseKey section of the registry. The function returns True or False indicating success. If the key already exists, the result is True.  **RegistryCreateValue**  Public Function RegistryCreateValue(BaseKey As Long, KeyName As String, \_  ValueName As String, ValueValue As Variant, \_  Optional CreateKeyIfNotExists As Boolean = False) As Boolean  This function creates a new value in the registry named ValueName in KeyName in BaseKey. If CreateKeyIfNotExists is True, the key named in KeyName is created if it does not exist. If the value named in ValueName already exists, its value is updated to the new value in ValueValue. This function returns True or False indicating success. ValueValue must be a String or Long type value.  **RegistryDeleteKey**  Public Function RegistryDeleteKey(BaseKey As Long, KeyName As String) As Boolean  This procedure deletes KeyName and all subkeys and values within KeyName. It returns True or False indicating success. If KeyName does not exist, the result is True.  **RegistryDeleteValue**  Public Function RegistryDeleteValue(BaseKey As Long, KeyName As String, ValueName As String) As Boolean  This procedure deletes the value named by ValueName from KeyName. The function returns True or False indicating success. If the value named by ValueName does not exist, the result is True.  **RegistryGetValue**  Public Function RegistryGetValue(BaseKey As Long, KeyName As String, \_  ValueName As String) As Variant  This function returns the value of the value named in ValueName of KeyName. It returns NULL if an error occurs. This function, along with the RegistryUpdateValue function, are the primary workers of these procedures. Most of what you need to do can be accomplished with these two functions.  **RegistsryGetValueType**  Public Function RegistryGetValueType(BaseKey As Long, KeyName As String, ValueName As String) As REG\_DATA\_TYPE  This function returns the data type of the Value stored in ValueName of KeyName. It will return either  REG\_INVALID = -1 (invalid type), REG\_SZ = 1 (String type) or  REG\_DWORD = 4 (Long type).  **RegistryKeyExists**  Public Function RegistryKeyExists(BaseKey As Long, KeyName As String, \_  Optional CreateIfNotExists As Boolean = False) As Boolean  This function returns True or False indicating whether the key named in KeyName exists. If the CreateIfNotExists parameter is True, the key will be created and the result will be True if the key was successfully created.  **RegistryUpdateValue**  Public Function RegistryUpdateValue(BaseKey As Long, KeyName As String, \_  ValueName As String, NewValue As Variant, Optional CreateIfNotExists As Boolean = True) As Boolean  This procedure updates the value of the existing value named by ValueName in KeyName with the new value NewValue. If value named in ValueName does not exist and CreateIfNotExists is True, the value is  create. If CreateIfNotExists is True and the key named in KeyName does not exist, then the key is create. Therefore, you can use this function to create new values in new keys, and the procedure will automatically create the new keys and values as required. This function returns True or False indicating success.  **RegistryValueExists**  Public Function RegistryValueExists(BaseKey As Long, KeyName As String, \_  ValueName As String, Optional CreateIfNotExists As Boolean = False, \_  Optional CreateType As REG\_DATA\_TYPE = REG\_DWORD) As Boolean  This function returns True or False indicating whether a registry value named ValueName exists in the key KeyName in BaseKey. If CreateIfNotExists is True, the key and/or the value is create if it does not exist. The CreateType parameter indicates whether to create a String type value (CreateType := REG\_SZ = 1) or a Long type value (CreateType := REG\_DWORD = 4).  The complete VBA code follows (not including the GetSystemErrorMessageText function, [available here](http://www.cpearson.com/excel/FormatMessage.htm))  Option Explicit  Option Compare Text  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' modRegistry  ' By Chip Pearson, www.cpearson.com, chip@cpearson.com  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  '  ' This function provides several functions related to working with keys and values in the system  ' registry. These routines call upon one another, so you should import this entire module into  ' your project rather than just copy/pasting an individual procedures.  '  ' This module is described and avaialable for download at http://www.cpearson.com/Excel/Registry.htm.  '  ' Error conditions and details are reported in the following public variables:  ' G\_Reg\_AppErrNum As Long Returns the module-defined error number.  ' G\_Reg\_AppErrText As String Returns the text description of G\_Reg\_AppErrNum  ' G\_Reg\_SysErrNum As Long Returns the system error number, usually the value of Err.LastDllError  ' G\_Reg\_SysErrText As String Returns the text description associated with G\_Reg\_SysErrNum, the text  ' returned from GetSystemErrorMessageText.  '  ' This module requires the moGetSystemErrorMessageText module, described and available for download at  ' http://www.cpearson.com/excel/FormatMessage.htm. This module itself is described and available for  ' download at http://www.cpearson.com/excel/registry.htm.  '  ' In all functions with a BaseKey parameter, the value of BaseKey must be either HKEY\_CURRENT\_USER (or HKCU) or  ' HKEY\_LOCAL\_MACHINE (or HKML). Any other value is invalid.  '  ' Public Functions In This Module:  ' --------------------------------  ' RegistryGetValue  ' RegistryGetValueType  ' RegistryCreateKey  ' RegistryCreateValue  ' RegistryDeleteKey  ' RegistryDeleteValue  ' RegistryKeyExists  ' RegistryValueExists  ' RegistryUpdateValue  '  ' See http://www.cpearson.com/excel/registry.htm for details about these procedures.  '  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Error Constants  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Public Const C\_REG\_ERR\_NO\_ERROR = 0  Public Const C\_REG\_ERR\_INVALID\_BASE\_KEY = vbObjectError + 1  Public Const C\_REG\_ERR\_INVALID\_DATA\_TYPE = vbObjectError + 2  Public Const C\_REG\_ERR\_KEY\_NOT\_FOUND = vbObjectError + 3  Public Const C\_REG\_ERR\_VALUE\_NOT\_FOUND = vbObjectError + 4  Public Const C\_REG\_ERR\_DATA\_TYPE\_MISMATCH = vbObjectError + 5  Public Const C\_REG\_ERR\_ENTRY\_LOCKED = vbObjectError + 6  Public Const C\_REG\_ERR\_INVALID\_KEYNAME = vbObjectError + 7  Public Const C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY = vbObjectError + 8  Public Const C\_REG\_ERR\_UNABLE\_TO\_READ\_KEY = vbObjectError + 9  Public Const C\_REG\_ERR\_UNABLE\_TO\_CREATE\_KEY = vbObjectError + 10  Public Const C\_REG\_ERR\_UBABLE\_TO\_READ\_VALUE = vbObjectError + 11  Public Const C\_REG\_ERR\_UNABLE\_TO\_UDPATE\_VALUE = vbObjectError + 12  Public Const C\_REG\_ERR\_UNABLE\_TO\_CREATE\_VALUE = vbObjectError + 13  Public Const C\_REG\_ERR\_UNABLE\_TO\_DELETE\_KEY = vbObjectError + 14  Public Const C\_REG\_ERR\_UNABLE\_TO\_DELETE\_VALUE = vbObjectError + 15  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' API Constants  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Public Const HKEY\_CURRENT\_USER As Long = &H80000001  Public Const HKEY\_LOCAL\_MACHINE As Long = &H80000002  Public Const HKEY\_CLASSES\_ROOT = &H80000000  Public Const HKEY\_CURRENT\_CONFIG = &H80000005  Public Const HKEY\_DYN\_DATA = &H80000006  Public Const HKEY\_PERFORMANCE\_DATA = &H80000004  Public Const HKEY\_USERS = &H80000003  Public Const HKCU = HKEY\_CURRENT\_USER  Public Const HKLM = HKEY\_LOCAL\_MACHINE  Private Const REGSTR\_MAX\_VALUE\_LENGTH As Long = &H100  Private Const KEY\_QUERY\_VALUE = &H1  Private Const KEY\_SET\_VALUE = &H2  Private Const KEY\_CREATE\_SUB\_KEY = &H4  Private Const KEY\_ENUMERATE\_SUB\_KEYS = &H8  Private Const KEY\_NOTIFY = &H10  Private Const KEY\_CREATE\_LINK = &H20  Private Const KEY\_ALL\_ACCESS = &H3F  Private Const REG\_CREATED\_NEW\_KEY = &H1  Private Const REG\_OPENED\_EXISTING\_KEY = &H2  Private Const STANDARD\_RIGHTS\_ALL = &H1F0000  Private Const SPECIFIC\_RIGHTS\_ALL = &HFFFF  Private Const REG\_OPTION\_NON\_VOLATILE = 0&  Private Const REG\_OPTION\_VOLATILE = &H1  Private Const ERROR\_SUCCESS = 0&  Private Const ERROR\_ACCESS\_DENIED = 5  Private Const ERROR\_INVALID\_DATA = 13&  Private Const ERROR\_MORE\_DATA = 234 ' dderror  Private Const ERROR\_NO\_MORE\_ITEMS = 259  Private Const S\_OK = &H0  Private Const MAX\_DATA\_BUFFER\_SIZE = 1024  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' API Types  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Private Type SECURITY\_ATTRIBUTES  nLength As Long  lpSecurityDescriptor As Long  bInheritHandle As Boolean  End Type  Private Type FILETIME  dwLowDateTime As Long  dwHighDateTime As Long  End Type  Public Enum REG\_DATA\_TYPE  REG\_INVALID = -1 ' Invalid  REG\_SZ = 1 ' String  REG\_DWORD = 4 ' Long  End Enum  Private Type ACL  AclRevision As Byte  Sbz1 As Byte  AclSize As Integer  AceCount As Integer  Sbz2 As Integer  End Type  Private Type SECURITY\_DESCRIPTOR  Revision As Byte  Sbz1 As Byte  Control As Long  Owner As Long  Group As Long  Sacl As ACL  Dacl As ACL  End Type  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' API Declares  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Private Declare Function RegCloseKey Lib "advapi32.dll" ( \_  ByVal HKey As Long) As Long  Private Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias "RegCreateKeyExA" ( \_  ByVal HKey As Long, \_  ByVal lpSubKey As String, \_  ByVal Reserved As Long, \_  ByVal lpClass As String, \_  ByVal dwOptions As Long, \_  ByVal samDesired As Long, \_  lpSecurityAttributes As SECURITY\_ATTRIBUTES, \_  phkResult As Long, \_  lpdwDisposition As Long) As Long  Private Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" ( \_  ByVal HKey As Long, \_  ByVal lpSubKey As String) As Long  Private Declare Function RegOpenKey Lib "advapi32.dll" Alias "RegOpenKeyA" ( \_  ByVal HKey As Long, \_  ByVal lpSubKey As String, \_  phkResult As Long) As Long  Private Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValueA" ( \_  ByVal HKey As Long, \_  ByVal lpValueName As String) As Long  Private Declare Function RegEnumKey Lib "advapi32.dll" Alias "RegEnumKeyA" ( \_  ByVal HKey As Long, \_  ByVal dwIndex As Long, \_  ByVal lpName As String, \_  ByVal cbName As Long) As Long  Private Declare Function RegEnumKeyEx Lib "advapi32.dll" Alias "RegEnumKeyExA" ( \_  ByVal HKey As Long, \_  ByVal dwIndex As Long, \_  ByVal lpName As String, \_  lpcbName As Long, \_  ByVal lpReserved As Long, \_  ByVal lpClass As String, \_  lpcbClass As Long, \_  lpftLastWriteTime As FILETIME) As Long  Private Declare Function RegEnumValue Lib "advapi32.dll" Alias "RegEnumValueA" ( \_  ByVal HKey As Long, \_  ByVal dwIndex As Long, \_  ByVal lpValueName As String, \_  lpcbValueName As Long, \_  ByVal lpReserved As Long, \_  lpType As Long, \_  lpData As Byte, \_  lpcbData As Long) As Long  Private Declare Function RegFlushKey Lib "advapi32.dll" ( \_  ByVal HKey As Long) As Long  Private Declare Function RegGetKeySecurity Lib "advapi32.dll" ( \_  ByVal HKey As Long, \_  ByVal SecurityInformation As Long, \_  pSecurityDescriptor As SECURITY\_DESCRIPTOR, \_  lpcbSecurityDescriptor As Long) As Long  Private Declare Function RegQueryInfoKey Lib "advapi32.dll" Alias "RegQueryInfoKeyA" ( \_  ByVal HKey As Long, \_  ByVal lpClass As String, \_  lpcbClass As Long, \_  ByVal lpReserved As Long, \_  lpcSubKeys As Long, \_  lpcbMaxSubKeyLen As Long, \_  lpcbMaxClassLen As Long, \_  lpcValues As Long, \_  lpcbMaxValueNameLen As Long, \_  lpcbMaxValueLen As Long, \_  lpcbSecurityDescriptor As Long, \_  lpftLastWriteTime As FILETIME) As Long  Private Declare Function RegQueryValue Lib "advapi32.dll" Alias "RegQueryValueA" ( \_  ByVal HKey As Long, \_  ByVal lpSubKey As String, \_  ByVal lpValue As String, \_  lpcbValue As Long) As Long  Private Declare Function RegQueryValueEx Lib "advapi32.dll" Alias "RegQueryValueExA" ( \_  ByVal HKey As Long, \_  ByVal lpValueName As String, \_  ByVal lpReserved As Long, \_  lpType As Long, \_  lpData As Any, \_  lpcbData As Long) As Long  Private Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" ( \_  ByVal HKey As Long, \_  ByVal lpValueName As String, \_  ByVal Reserved As Long, \_  ByVal dwType As Long, \_  lpData As Any, \_  ByVal cbData As Long) As Long  Private Declare Function RegSetValueExStr Lib "advapi32" Alias "RegSetValueExA" ( \_  ByVal HKey As Long, \_  ByVal lpValueName As String, \_  ByVal Reserved As Long, \_  ByVal dwType As Long, \_  ByVal szData As String, \_  ByVal cbData As Long) As Long  Private Declare Function RegSetValueExLong Lib "advapi32" Alias "RegSetValueExA" ( \_  ByVal HKey As Long, \_  ByVal lpValueName As String, \_  ByVal Reserved As Long, \_  ByVal dwType As Long, \_  szData As Long, \_  ByVal cbData As Long) As Long  Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" ( \_  ByVal HKey As Long, \_  ByVal lpSubKey As String, \_  ByVal ulOptions As Long, \_  ByVal samDesired As Long, \_  phkResult As Long) As Long  Private Declare Function RegQueryValueExStr Lib "advapi32" Alias "RegQueryValueExA" ( \_  ByVal HKey As Long, \_  ByVal lpValueName As String, \_  ByVal lpReserved As Long, \_  ByRef lpType As Long, \_  ByVal szData As String, \_  ByRef lpcbData As Long) As Long  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Application Constants  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Public Type RegValue  ValueName As String  ValueValue As Variant  End Type  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Public Variables  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Public G\_Reg\_AppErrNum As Long  Public G\_Reg\_AppErrText As String  Public G\_Reg\_SysErrNum As Long  Public G\_Reg\_SysErrText As String  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Private Variables  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Public Functions  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Public Function RegistryGetValue(BaseKey As Long, KeyName As String, \_  ValueName As String) As Variant  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryGetValue  ' This funciton gets the value of of the specified ValueName in the  ' key KeyName in the base key BaseKey. Returns NULL if an error  ' occurred.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim HKey As Long  Dim Res As Long  Dim RegDataType As REG\_DATA\_TYPE  Dim LenData As Long  Dim LongData As Long  Dim StringData As String  Dim IntArr(0 To 1024) As Integer  Dim LenStringData As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValue = Null  Exit Function  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValue = Null  Exit Function  End If  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_KEY\_NOT\_FOUND  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValue = Null  Exit Function  End If  RegDataType = RegistryGetValueType(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName)  HKey = OpenRegistryKey(BaseKey:=BaseKey, KeyName:=KeyName)  If HKey = 0 Then  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=Res)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValue = Null  Exit Function  End If  If RegDataType = REG\_DWORD Then  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Data is Long data-type.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Res = RegQueryValueEx(HKey:=HKey, lpValueName:=ValueName, lpReserved:=0&, \_  lpType:=RegDataType, lpData:=LongData, lpcbData:=Len(LongData))  If Res = ERROR\_SUCCESS Then  RegistryGetValue = LongData  Exit Function  Else  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=Res)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UBABLE\_TO\_READ\_VALUE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegCloseKey HKey  RegistryGetValue = Null  Exit Function  End If  ElseIf RegDataType = REG\_SZ Then  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Data is String data-type.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  StringData = String$(MAX\_DATA\_BUFFER\_SIZE, vbNullChar)  LenStringData = Len(StringData)  Res = RegQueryValueExStr(HKey:=HKey, lpValueName:=ValueName, lpReserved:=0&, \_  lpType:=RegDataType, szData:=StringData, lpcbData:=LenStringData)  If Res <> ERROR\_SUCCESS Then  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=Res)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UBABLE\_TO\_READ\_VALUE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegCloseKey HKey  RegistryGetValue = Null  Exit Function  End If  StringData = TrimToNull(StringData)  RegistryGetValue = StringData  Else  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_DATA\_TYPE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValue = Null  End If  End Function  Public Function RegistryKeyExists(BaseKey As Long, KeyName As String, \_  Optional CreateIfNotExists As Boolean = False) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryKeyExists  ' Returns True or False indicating whether KeyName exists in BaseKey.  ' Returns False if an error occurred. See the global error values  ' for more information. If CreateIfNotExists is True and the  ' key does not exist, it will be created.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim HKey As Long  Dim Res As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryKeyExists = False  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryKeyExists = False  End If  Res = RegOpenKey(HKey:=BaseKey, lpSubKey:=KeyName, phkResult:=HKey)  If Res = ERROR\_SUCCESS Then  RegistryKeyExists = True  Else  RegistryKeyExists = False  If CreateIfNotExists = True Then  Res = RegistryCreateKey(BaseKey:=BaseKey, KeyName:=KeyName)  RegistryKeyExists = CBool(Res)  End If  End If  RegCloseKey HKey:=HKey  End Function  Public Function RegistryValueExists(BaseKey As Long, KeyName As String, \_  ValueName As String, Optional CreateIfNotExists As Boolean = False, \_  Optional CreateType As REG\_DATA\_TYPE = REG\_DWORD) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryValueExists  ' This returns True or False indicating whether ValueName exists in  ' KeyName in BaseKey.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim HKey As Long  Dim Res As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryValueExists = False  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryValueExists = False  End If  HKey = OpenRegistryKey(BaseKey:=BaseKey, KeyName:=KeyName)  If HKey = 0 Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryValueExists = False  End If  Res = RegQueryValueEx(HKey:=HKey, lpValueName:=ValueName, lpReserved:=0&, lpType:=0&, lpData:=0&, lpcbData:=0&)  If (Res = ERROR\_SUCCESS) Or (Res = ERROR\_MORE\_DATA) Then  RegistryValueExists = True  Else  If CreateIfNotExists = True Then  If CreateType = REG\_DWORD Then  Res = RegistryCreateValue(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName, \_  ValueValue:=0&, CreateKeyIfNotExists:=True)  Else  Res = RegistryCreateValue(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName, \_  ValueValue:=vbNullString, CreateKeyIfNotExists:=True)  End If  If CBool(Res) = True Then  RegistryValueExists = True  Else  RegistryValueExists = False  End If  End If  End If  RegCloseKey HKey  End Function  Public Function RegistryGetValueType(BaseKey As Long, KeyName As String, ValueName As String) As REG\_DATA\_TYPE  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryGetValueType  ' This returns the data type of value named in ValueName. The procedures in  ' this module support only Longs and Strings, so the result will be REG\_SZ  ' for a string, REG\_DWORD for a Long, or REG\_INVALID for any other data type.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Res As Long  Dim HKey As Long  Dim DataType As REG\_DATA\_TYPE  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValueType = False  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValueType = False  End If  Res = RegOpenKey(HKey:=BaseKey, lpSubKey:=KeyName, phkResult:=HKey)  If Res <> ERROR\_SUCCESS Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValueType = REG\_INVALID  Exit Function  End If    Res = RegQueryValueEx(HKey:=HKey, lpValueName:=ValueName, lpReserved:=0&, lpType:=DataType, lpData:=0&, lpcbData:=0&)  If (Res <> ERROR\_SUCCESS) And (Res <> ERROR\_MORE\_DATA) Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_UBABLE\_TO\_READ\_VALUE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryGetValueType = REG\_INVALID  RegCloseKey HKey  Exit Function  End If    Select Case DataType  Case REG\_SZ  RegistryGetValueType = REG\_SZ  Case REG\_DWORD  RegistryGetValueType = REG\_DWORD  Case Else  RegistryGetValueType = REG\_INVALID  End Select  RegCloseKey HKey  End Function  Public Function RegistryCreateValue(BaseKey As Long, KeyName As String, \_  ValueName As String, ValueValue As Variant, \_  Optional CreateKeyIfNotExists As Boolean = False) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryCreateValue  ' This creates a value named ValueName in KeyName in BaseKey with a value  ' of ValueValue. If the key named by KeyName does not exist, and  ' CreateKeyIfNotExist is True, the key will be created. If the value  ' already exists, its value is set to the new value if they are  ' compatible data types.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim HKey As Long  Dim Res As Long  Dim DataType As REG\_DATA\_TYPE  Dim StringValue As String  Dim LongValue As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  End If  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName, \_  CreateIfNotExists:=CreateKeyIfNotExists) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_KEY\_NOT\_FOUND  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  End If      If IsCompatibleValueValue(Var:=ValueValue) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_DATA\_TYPE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  End If  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName, CreateIfNotExists:=False) = False Then  If CreateKeyIfNotExists = True Then  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName, CreateIfNotExists:=True) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_CREATE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  End If  Else  G\_Reg\_AppErrNum = C\_REG\_ERR\_KEY\_NOT\_FOUND  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  End If  End If  If RegistryValueExists(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName) = True Then  DataType = RegistryGetValueType(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName)  If DataType = REG\_SZ Then  If VarType(ValueValue) <> vbString Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_DATA\_TYPE\_MISMATCH  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  Exit Function  Else  '''''''''''''''''''''''''''''  ' ValueValue is a string. OK.  '''''''''''''''''''''''''''''  End If  Else  '''''''''''''''''''''''''  ' ValueValue is numeric  '''''''''''''''''''''''''  End If  Else  '''''''''''''''''''''''  ' Value does not exist.  ' Set the DataType.  '''''''''''''''''''''''  If VarType(ValueValue) = vbString Then  DataType = REG\_SZ  Else  DataType = REG\_DWORD  End If  End If  If DataType = REG\_DWORD Then  LongValue = CLng(ValueValue)  HKey = OpenRegistryKey(BaseKey:=BaseKey, KeyName:=KeyName)  If HKey = 0 Then  G\_Reg\_SysErrNum = Err.LastDllError  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=G\_Reg\_SysErrNum)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegCloseKey HKey  RegistryCreateValue = False  Exit Function  End If    Res = RegSetValueExLong(HKey:=HKey, lpValueName:=ValueName, Reserved:=0&, \_  dwType:=REG\_DWORD, szData:=LongValue, cbData:=Len(LongValue))  If Res <> ERROR\_SUCCESS Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_UDPATE\_VALUE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegCloseKey HKey  RegistryCreateValue = False  Exit Function  End If  Else  StringValue = CStr(ValueValue)  HKey = OpenRegistryKey(BaseKey:=BaseKey, KeyName:=KeyName)  If HKey = 0 Then  G\_Reg\_SysErrNum = Err.LastDllError  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=G\_Reg\_SysErrNum)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegCloseKey HKey  RegistryCreateValue = False  Exit Function  End If  Res = RegSetValueExStr(HKey:=HKey, lpValueName:=ValueName, Reserved:=0&, \_  dwType:=REG\_SZ, szData:=StringValue, cbData:=Len(StringValue))  If Res <> ERROR\_SUCCESS Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_UDPATE\_VALUE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateValue = False  RegCloseKey HKey  Exit Function  End If  End If  RegCloseKey HKey  RegistryCreateValue = True  End Function  Public Function RegistryCreateKey(BaseKey As Long, KeyName As String) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryCreateKey  ' This function creates a Key named KeyName in BaseKey. Returns True if successful  ' or False if an error occurred.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Res As Long  Dim HKey As Long  Dim DataType As REG\_DATA\_TYPE  Dim SecAttrib As SECURITY\_ATTRIBUTES  Dim Disposition As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateKey = False  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateKey = False  End If  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName) = True Then  '''''''''''''''''''''''''''  ' Key already exist. Return  ' True as if we created it.  '''''''''''''''''''''''''''  RegistryCreateKey = True  Exit Function  End If  Res = RegCreateKeyEx(HKey:=BaseKey, lpSubKey:=KeyName, Reserved:=0&, lpClass:=vbNullString, \_  dwOptions:=REG\_OPTION\_NON\_VOLATILE, samDesired:=KEY\_ALL\_ACCESS, \_  lpSecurityAttributes:=SecAttrib, phkResult:=HKey, lpdwDisposition:=Disposition)  If Res <> ERROR\_SUCCESS Then  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=G\_Reg\_SysErrNum)  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryCreateKey = False  Exit Function  End If  RegistryCreateKey = True  End Function  Public Function RegistryDeleteValue(BaseKey As Long, KeyName As String, ValueName As String) As Boolean  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryDeleteValue  ' This deletes a value in KeyName in BaseKey. Returns True or False indicating  ' success. The function returns True if the Value does not exist.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Res As Long  Dim HKey As Long  Dim DataType As REG\_DATA\_TYPE  Dim SecAttrib As SECURITY\_ATTRIBUTES  Dim Disposition As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryDeleteValue = False  Exit Function  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryDeleteValue = False  Exit Function  End If  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName, CreateIfNotExists:=False) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_KEY\_NOT\_FOUND  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryDeleteValue = False  Exit Function  End If  HKey = OpenRegistryKey(BaseKey:=BaseKey, KeyName:=KeyName)  If HKey = 0 Then  RegistryDeleteValue = False  Exit Function  End If  If RegistryValueExists(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName) = False Then  RegCloseKey HKey  RegistryDeleteValue = True  Exit Function  End If  Res = RegDeleteValue(HKey:=HKey, lpValueName:=ValueName)  If Res <> ERROR\_SUCCESS Then  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=Res)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_DELETE\_VALUE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegCloseKey HKey  RegistryDeleteValue = False  Exit Function  End If    RegCloseKey HKey  RegistryDeleteValue = True  End Function  Public Function RegistryDeleteKey(BaseKey As Long, KeyName As String) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryDeleteKey  ' This delete the registry key named in KeyName in BaseKey. All subkeys and  ' values are deleted. Returns True or False indicating success. Returns  ' True if the key does not exist.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Res As Long  Dim HKey As Long  Dim DataType As REG\_DATA\_TYPE  Dim SecAttrib As SECURITY\_ATTRIBUTES  Dim Disposition As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryDeleteKey = False  Exit Function  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryDeleteKey = False  Exit Function  End If  If RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName, CreateIfNotExists:=False) = False Then  RegistryDeleteKey = True  Exit Function  End If  HKey = OpenRegistryKey(BaseKey:=BaseKey, KeyName:=KeyName)  If HKey = 0 Then  RegistryDeleteKey = False  Exit Function  End If  Res = RegDeleteKey(HKey:=BaseKey, lpSubKey:=KeyName)  RegCloseKey HKey  If Res <> ERROR\_SUCCESS Then  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(Res)  G\_Reg\_AppErrNum = C\_REG\_ERR\_UNABLE\_TO\_DELETE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryDeleteKey = False  Exit Function  End If  RegistryDeleteKey = True    End Function  Public Function RegistryUpdateValue(BaseKey As Long, KeyName As String, \_  ValueName As String, NewValue As Variant, Optional CreateIfNotExists As Boolean = True) As Boolean  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' RegistryUpdateValue  ' This updates the value of a key. It calls RegistryDeleteValue to delete the  ' value and the RegistryCreateValue to re-create the value. Returns True or  ' False indicating success.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Res As Boolean  Dim HKey As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey:=BaseKey) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryUpdateValue = False  Exit Function  End If  If IsValidKeyName(KeyName:=KeyName) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryUpdateValue = False  Exit Function  End If  If IsCompatibleValueValue(Var:=NewValue) = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_DATA\_TYPE  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryUpdateValue = False  Exit Function  End If  Res = RegistryKeyExists(BaseKey:=BaseKey, KeyName:=KeyName, CreateIfNotExists:=True)  If Res = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_KEY\_NOT\_FOUND  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryUpdateValue = False  Exit Function  End If  If VarType(NewValue) = vbString Then  Res = RegistryValueExists(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName, \_  CreateIfNotExists:=CreateIfNotExists, CreateType:=REG\_DWORD)  Else  Res = RegistryValueExists(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName, \_  CreateIfNotExists:=CreateIfNotExists, CreateType:=REG\_SZ)  End If  If Res = False Then  G\_Reg\_AppErrNum = C\_REG\_ERR\_VALUE\_NOT\_FOUND  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  RegistryUpdateValue = False  Exit Function  End If  Res = RegistryDeleteValue(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName)  Res = RegistryCreateValue(BaseKey:=BaseKey, KeyName:=KeyName, ValueName:=ValueName, ValueValue:=NewValue, CreateKeyIfNotExists:=True)  RegistryUpdateValue = Res  End Function  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' Private Functions  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Private Function OpenRegistryKey(BaseKey As Long, KeyName As String) As Long  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' OpenRegistryKey  ' This opens the KeyName in BaseKey and returns the key handle  ' if successful or 0 if an error occurred.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Res As Long  Dim HKey As Long  ResetErrorVariables  If IsValidBaseKey(BaseKey) = False Then  ''''''''''''''''''''''''''''''''''''''  ' Invalid Base Key. Return 0 and  ' get out.  ''''''''''''''''''''''''''''''''''''''  OpenRegistryKey = 0  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  Exit Function  End If  Res = RegOpenKeyEx(HKey:=BaseKey, lpSubKey:=KeyName, ulOptions:=0&, samDesired:=KEY\_ALL\_ACCESS, phkResult:=HKey)  If Res <> ERROR\_SUCCESS Then  OpenRegistryKey = 0  G\_Reg\_SysErrNum = Res  G\_Reg\_SysErrText = GetSystemErrorMessageText(ErrorNumber:=Res)  G\_Reg\_AppErrNum = C\_REG\_ERR\_INVALID\_BASE\_KEY  G\_Reg\_AppErrText = GetAppErrText(G\_Reg\_AppErrNum)  Exit Function  End If  OpenRegistryKey = HKey  End Function  Private Function TrimToNull(Text As String, \_  Optional Reverse As Boolean = False) As String  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' TrimToNull  ' If Reverse is omitted or False, the function returns the  ' portion of Text that is to the left of the first vbNullChar  ' character. The vbNullChar is not returned. If Reverse is  ' True, the function returns the portion to the left of the  ' last vbNullChar. The vbNullChar is not returned. In either  ' case, if vbNullChar is not found, the entire string Text  ' is returned.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Pos As Long  If Reverse = False Then  Pos = InStr(1, Text, vbNullChar, vbTextCompare)  Else  Pos = InStrRev(Text, vbNullChar, -1, vbTextCompare)  End If  If Pos Then  TrimToNull = Left(Text, Pos - 1)  Else  TrimToNull = Text  End If  End Function  Private Function TrimToChar(Text As String, Char As String, \_  Optional ByVal Reverse As Boolean = False, \_  Optional ByVal CompareMode As VbCompareMethod) As String  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' If Reverse is False, the function returns the portion of  ' Text that is to the left of the first occurrence of Char.  ' If Reverse is True, the function returns the portion of  ' Text that is to the left of the last occurrence of Char.  ' If Char is not found, the entire string Text is returned.  ' If CompareMode is vbBinaryCompare, text is compared in  ' a CASE-SENSITIVE manner ("A"<>"a"). If CompareMode is any  ' other value, text is compared in CASE-INSENSITIVE mode ("A" = "a").  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Dim Pos As Long  If CompareMode <> vbBinaryCompare Then  CompareMode = vbTextCompare  End If  If Reverse = False Then  Pos = InStr(1, Text, Char, CompareMode)  Else  Pos = InStrRev(Text, Char, -1, CompareMode)  End If  If Pos Then  TrimToChar = Left(Text, Pos - 1)  Else  TrimToChar = Text  End If  End Function  Private Function IsValidBaseKey(BaseKey As Long) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' IsValidBaseKey  ' This returns True of BaseKey is valid base key  ' (HKEY\_CURRENT\_USER etc) or False if BaseKey is not  ' a valid base key.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Select Case BaseKey  Case HKEY\_CURRENT\_USER, HKEY\_LOCAL\_MACHINE, \_  HKEY\_CLASSES\_ROOT, HKEY\_CURRENT\_CONFIG, HKEY\_DYN\_DATA, \_  HKEY\_PERFORMANCE\_DATA, HKEY\_USERS  IsValidBaseKey = True  Case Else  IsValidBaseKey = False  End Select  End Function  Private Sub ResetErrorVariables()  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' ResetErrorVariables  ' This resets the global error values to their default  ' (no error) values.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  G\_Reg\_AppErrNum = C\_REG\_ERR\_NO\_ERROR  G\_Reg\_AppErrText = vbNullString  G\_Reg\_SysErrNum = C\_REG\_ERR\_NO\_ERROR  G\_Reg\_SysErrText = vbNullString  End Sub  Private Function GetAppErrText(ErrNum As Long) As String  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' GetAppErrText  ' This returns the text description of the application error  ' number in ErrNum.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Select Case ErrNum  Case C\_REG\_ERR\_NO\_ERROR  GetAppErrText = vbNullString  Case C\_REG\_ERR\_INVALID\_BASE\_KEY  GetAppErrText = "Invalid Base Key Value."  Case C\_REG\_ERR\_INVALID\_DATA\_TYPE  GetAppErrText = "Invalid Data Type."  Case C\_REG\_ERR\_KEY\_NOT\_FOUND  GetAppErrText = "Key Not Found."  Case C\_REG\_ERR\_VALUE\_NOT\_FOUND  GetAppErrText = "Value Not Found."  Case C\_REG\_ERR\_DATA\_TYPE\_MISMATCH  GetAppErrText = "Value Data Type Mismatch."  Case C\_REG\_ERR\_ENTRY\_LOCKED  GetAppErrText = "Registry Entry Locked."  Case C\_REG\_ERR\_INVALID\_KEYNAME  GetAppErrText = "The Specified Key Is Invalid."  Case C\_REG\_ERR\_UNABLE\_TO\_OPEN\_KEY  GetAppErrText = "Unable To Open Key."  Case C\_REG\_ERR\_UNABLE\_TO\_READ\_KEY  GetAppErrText = "Unable To Read Key."  Case C\_REG\_ERR\_UNABLE\_TO\_CREATE\_KEY  GetAppErrText = "Unable To Create Key."  Case C\_REG\_ERR\_UBABLE\_TO\_READ\_VALUE  GetAppErrText = "Unable To Read Value."  Case C\_REG\_ERR\_UNABLE\_TO\_UDPATE\_VALUE  GetAppErrText = "Unable To Update Value."  Case C\_REG\_ERR\_UNABLE\_TO\_CREATE\_VALUE  GetAppErrText = "Unable To Create Value."  Case C\_REG\_ERR\_UNABLE\_TO\_DELETE\_KEY  GetAppErrText = "Unable To Delete Key."  Case C\_REG\_ERR\_UNABLE\_TO\_DELETE\_VALUE  GetAppErrText = "Unable To Delete Value."          Case Else  GetAppErrText = "Undefined Error."  End Select  End Function  Private Function IsStringValidLength(Text As String) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' IsStringValidLength  ' This tests whether the length of Text is less than  ' REGSTR\_MAX\_VALUE\_LENGTH.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  IsStringValidLength = (Len(Text) <= REGSTR\_MAX\_VALUE\_LENGTH)  End Function  Private Function IsValidKeyName(KeyName As String) As Boolean  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' IsValidKeyName  ' Returns True or False indicating whether KeyName is valid.  ' An invalid key is one whose name length is greater than  ' REGSTR\_MAX\_VALUE\_LENGTH or is all spaces or is an empty string.  '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  IsValidKeyName = (Len(KeyName) <= REGSTR\_MAX\_VALUE\_LENGTH) And (Len(Trim(KeyName)) > 0)  End Function  Private Function IsValidDataType(DataType As REG\_DATA\_TYPE) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' IsValidDataType  ' This returns True or False indicating whether DataType is  ' a valid data type (REG\_SZ or REG\_DWORD).  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  Select Case DataType  Case REG\_SZ, REG\_DWORD  IsValidDataType = True  Case Else  IsValidDataType = False  End Select  End Function  Private Function IsCompatibleValueValue(Var As Variant) As Boolean  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  ' IsCompatibleValueValue  ' This test the VarType of Var to see if it is valid to be used  ' as a registry key value. Note that all numeric data types (Singles,  ' Doubles, etc) are considered value, even though their values will  ' be changed when converted to longs.  ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''  If VarType(Var) >= vbArray Then  IsCompatibleValueValue = False  Exit Function  End If  If IsArray(Var) = True Then  IsCompatibleValueValue = False  Exit Function  End If  If IsObject(Var) = True Then  IsCompatibleValueValue = False  Exit Function  End If  Select Case VarType(Var)  Case vbBoolean, vbByte, vbCurrency, vbDate, vbDouble, vbInteger, vbLong, vbSingle, vbString  IsCompatibleValueValue = True  Case Else  IsCompatibleValueValue = False  End Select  End Function |  |