User Manual and Source Code
for
Email Contact Import Application
developed
For Driving Development of Maidenhead

Written by: Nigel Kitcher

28th January 2002

1. Project Specification

1.1 Company and Project Overview

Driving Development is a driver training company with the objectives of teaching limit handling whether at speed or under wet conditions.

The success of the company is largely due to the growth of its Internet presence as its client base is largely made up of IT professionals. Enquiries and contact information comes into the company via the email system either as the output from the enquiry form on the web site, or from periodical news and discussion group digests.

There is a requirement to take the data from these emailed sources and to add them to the contacts held within the email system.

1.2 Detail

The current emails that arrive are in one of two formats.

The first of these (the enquiry form format) is simply one email per contact with the contacts name and email address presented within opening and closing square brackets.

For example:

```
Name [Stephen Whitehorn]       email address [s.whitehorn@talk21.com]
```

These emails can distinguished by the subject line of:
"*Enquiry from www.DrivingTechniques.co.uk*"

The second format is that used by the periodical news and discussion group digests. These are presented by the contact name, which is enclosed in double quotes, and the email address, which is enclosed in less than and greater than signs.

For example:

```
"nes"       email address <nes@globalnet.co.uk>
```

These emails have no constant distinguishing email subject line, as they will typically include a date or a digest number. As there are a number of sources the program should be flexible to allow a specified subject line, which will be used as a basis for pattern matching.

Outlook is the current company email system and a number of subfolders have been created to allow efficient filing of emails and contacts. As such the program should allow the user to specify the location of the emails to be scanned and the location in which to place the contacts once created.

1.3 Summary of Requirements

- Should be able to graphically select the source folder for emails to scan, and also the destination folder in which to place contacts
- Should be able to filter emails for checking by pattern matching text in the subject line
- Should be able to select from a range of common text delimiters either square brackets or double quotes and angle brackets.
- Contacts should not be duplicated if already existing in the system
- For auditing purposes it should be possible to specify a user defined message to be placed in the contacts notes during its creation
- User input should be saved and recovered for later sessions
- A user set up program should be provided for ease of installation
- A report should be generated of how many emails were checked, how many contacts were created, as well as any errors such as duplicate contacts

2. User Manual

2.1 Installation

To start the installation process double click the SETUP.EXE which maybe found on the setup disks.

The user is presented with Welcome dialog shown in Figure 1 below. Simply click [OK] to continue.

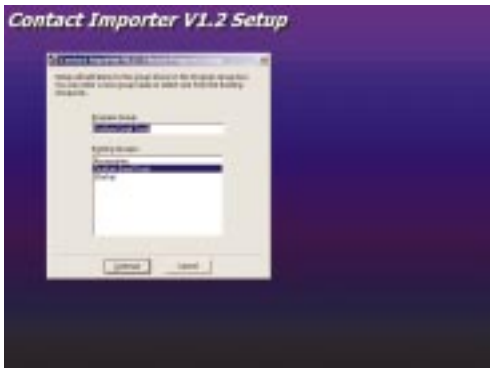
Figure 1 - Contact Importer Setup Dialog

Next the user is presented with the dialog show below to choose the installation path. This is typically "`C:\Program Files\DryBum Email Tools\`". Unless you have a requirement to change the local click the large [SETUP] button.

Figure 2 - Installation Folder Selection

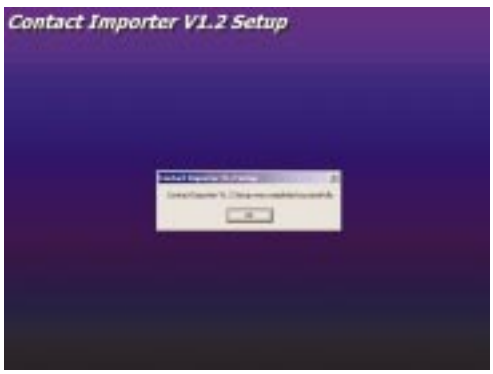
The next dialog is the Program Group Selection. The default group is "DryBum Email Tools". If there is no requirement to change this title, or add the application to an existing group simply choose [CONTINUE].

Figure 3 - Program Group Selection



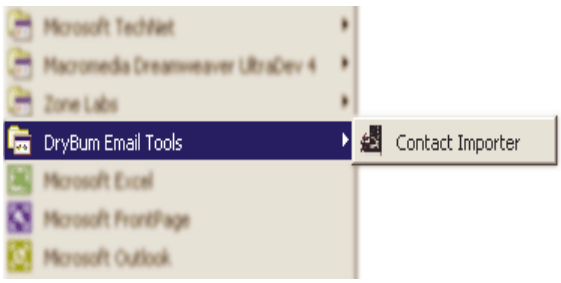
Once the files have been copied to the PC and the installation the user will see the screen shown in Figure 4 below. Choose [OK] and continue with section 2.2 for instructions on how to use the application to import contact details into Outlook.

Figure 4 - Installation Complete



2.2 Starting the Contact Importer Application

To start the Importer program choose the [Contact Importer] icon from the [Drybum Email Tools] program group on the Programs menu.



You will be presented with the About dialog box shown in Figure 6. This shows the version of the program together with hyperlinks to web sites of Driving Development and of Nkitcher. If you wish to display this dialog box once the program is running it may be found under “About” on the system menu of the main dialog, or alternatively click the “Outlook” logo on the main dialog.

Choose [OK] to continue to skip the About box.

Figure 6 - About Dialog

2.3 Main Dialog

The Contact Importer consists of the dialog box shown in Figure 7.

There are just two options. The first button ([Import]) starts the import process using the saved settings from a previous session.

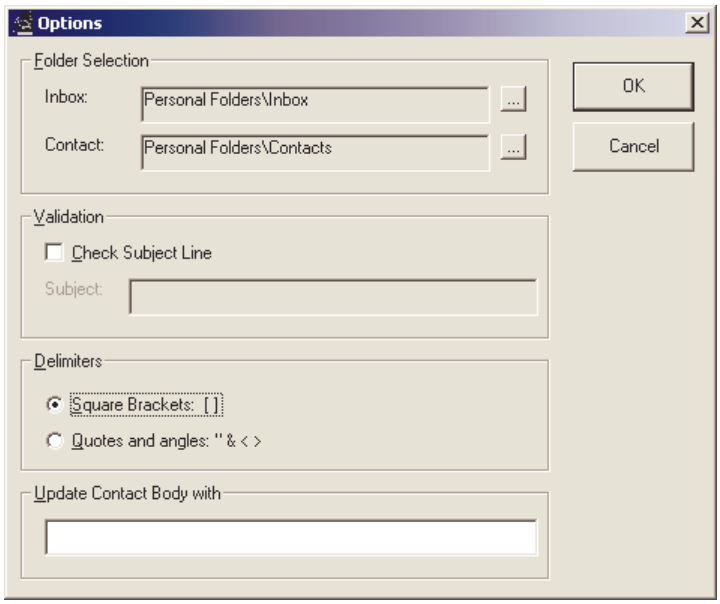
The second button ([Options]) allows the user to specify the settings such as the location of the incoming emails and where created contacts should be placed.

Figure 7 - Main Dialog

2.4 Setting Importation Settings

Choosing [Options] from the main dialog will display the Options Dialog box shown in Figure 8. This dialog is broken up into four sections.

Figure 8 - Options Dialog

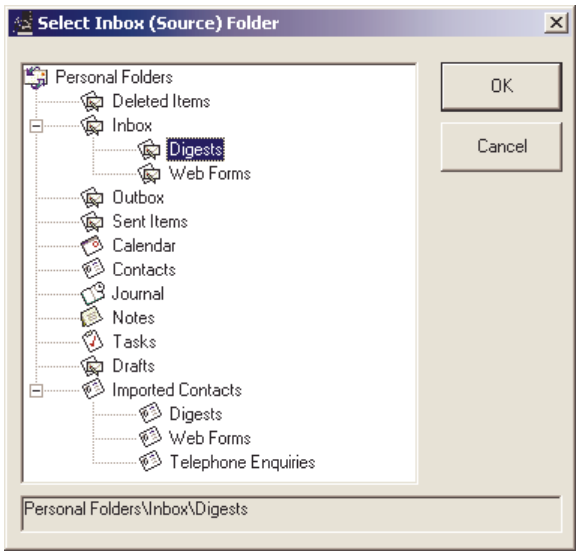


2.5 Folder Selection

The Folder Selection section allows the user to choose the Outlook folders to be used for scanning for emails and for creating contact records that it creates from the mails.

Selecting the ellipse button ([…]) for either the Inbox or Contact Folder will display the appropriate selection dialog. The Inbox folder selection dialog is shown below in Figure 9. Simply choose the required folder from the Outlook folder tree to change the selection shown at the bottom of the dialog and choose [OK]. Choosing [Cancel] will leave the folder unchanged.

Figure 9 - Outlook Folder Selection Dialog



2.5.1 Validation Section

The Validation section, shown in Figure 10 below, allows the scanned emails to be filtered based on the subject line of the email. This can be used to speed up processing where emails are not easily separated by filters within Outlook.

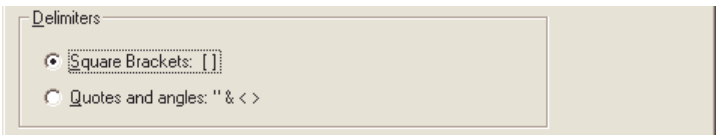
To enable this feature tick the “Check Subject Line” check box and enter the text to be matched in the “Subject” edit box. Note that the text entered simply has to appear anywhere within the subject line of the email. A complete match is not required.

Figure 10 - Validation section

2.5.2 Delimiters Section

The delimiters section allows the user to choose between square brackets or quotes and angle brackets. Simply choose the appropriate radio button as shown in Figure 11 below.

Figure 11 - Delimiters section



2.5.3 Update Contact Body Section

The “Update Contact Body” section permits the entry of free text that should be added to the body of any created contact. This is useful, for example, for adding the date or other reference to aid further processing within Outlook.

If the edit box is left blank the user is prompted every time an imported is performed to allow the body to be updated without having to select [Options] every time from the main dialog.

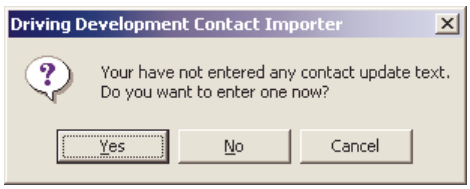
Figure 12 - Update Contact Body section

2.6 Performing an Import

Once the application has been configured the [Import] button on the main dialog can be used to start the process.

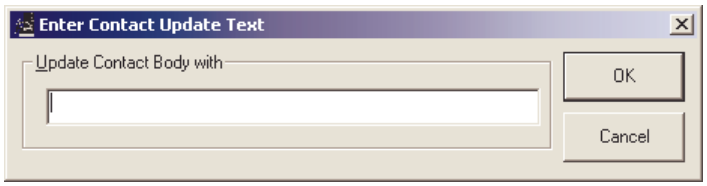
If the “Update Contact Body” edit box was left blank in the Options dialog the user is first offered, via the message box shown in Figure 13, the choice of entering text for this import run.

Figure 13 - Contact Update Text Reminder Message



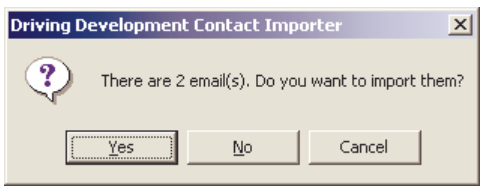
Choosing [Yes] will present the user with the dialog shown in Figure 14, whilst [No] will continue the run but not place any text within any new contacts body field. Selecting [Cancel] will abort the import.

Figure 14 - Contact Update Text Dialog



The user is next presented with confirmation of the number of emails that have been found in the selected input folder. The user must select [Yes] to continue

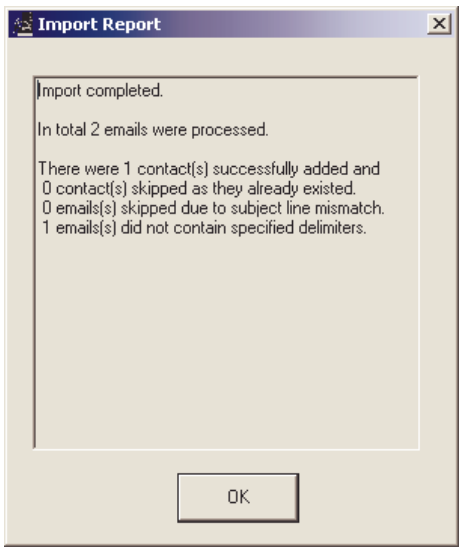
Figure 15 - Import Message box



Once the email scan has been completed a report will be displayed similar to that shown in Figure 16. This displays the total number of emails that were processed, the number of new contacts created, the number of duplicate contacts that were skipped, the number of emails that failed the subject line test (if selected), and the number of emails that did not contain contact information.

Simply choose [OK] to return to the main dialog.

Figure 16 - Import Report Dialog



3. **Source Code** (abridged)

3.1 frmAbout

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

Private Sub OK_Click()
'
'   Function:   OK_Click()
'
'   Parameters: none
'
'   Purpose:    Hides the About box
'
'   Updates:    24-Jan-02

    frmAbout.Hide

End Sub

Private Sub webDriving_Click()
'
'   Function:   webDriving_Click()
'
'   Parameters: none
'
'   Purpose:    Starts the web browser to point at the clients site when
'               the link Is clicked
'
'   Updates:    24-Jan-02

   Dim r As Long
   r = ShellExecute(0, "open", "http://www.drivingdevelopment.co.uk", 0, 0, 1)

End Sub

Private Sub webNKitcher_Click()
'
'   Function:   webNKitcher_Click()
'
'   Parameters: none
'
'   Purpose:    Starts the web browser to point at my web site when
'               the link Is clicked
'
'   Updates:    24-Jan-02

   Dim r As Long
   r = ShellExecute(0, "open", "http://www.nkitcher.co.uk", 0, 0, 1)

End Sub
```

3.2 frmContactUpdate

```
Private Sub Cancel_Click()
'
'   Function:   Cancel_Click()
'
'   Parameters: none
'
'   Purpose:    Closes the Contact update box
'
'   Updates:    24-Jan-02

    frmContactUpdate.Hide

End Sub

Private Sub OK_Click()
'
'   Function:   OK_Click()
'
'   Parameters: none
'
'   Purpose:    Get the user entered text And closes
'               contact update box
'
'   Updates:    24-Jan-02

    ' Retrieve the Text that will form the body text of the New contacts
    szContactUpdate = txtContactUpdate.Text

    frmContactUpdate.Hide
End Sub
```

3.3 frmFolderSelection

```
Option Explicit
Dim szUndoPath As String ' Remember the initial path For Cancelling this dialog

Private Sub Form_Load()
'
'   Function:   Form_Load()
'
'   Parameters: none
'
'   Purpose:
'               Gets the root of the MAPI namespace And generates
'               the tree view control of that namespace when the "FolderSelection"
'               form Is loaded.
'
'   Updates:    24-Jan-02

    Dim ol As Outlook.Application
    Dim olns As Outlook.NameSpace
    Dim objRootFolder As Outlook.MAPIFolder

    ' Get the Application Object.
    Set ol = New Outlook.Application

    ' Get the Namespace Object.
    Set olns = ol.GetNamespace("MAPI")

    ' Get the root of the default folders
    Set objRootFolder = olns.GetDefaultFolder(olFolderInbox).Parent

    ' Populate the TreeView control
    AddLeaf objRootFolder, OutlookFolderView, "", 1

    szUndoPath = lblPath.Caption

End Sub
```

```
Private Sub AddLeaf(ObjRoot As Outlook.MAPIFolder, vntTree As Variant, szParentCod
e As String, n As Integer)
'
'   Function:   AddLeaf
'
'   Parameters: ObjRoot         -   the root point For these nodes to be added at
'               vntTree         -   the main tree structure
'               szParentCode    -   the unique index of the parent of this node
'               n               -   the peer node counter For this level
'
'   Purpose:
'               Recursively iterates through the tree adding nodes And
'               child nodes to represent the MAPI name space.
'
'   Updates:    24-Jan-02


    Dim szNodeIndex As String
    Dim i As Integer
    Dim nodX As MSComctlLib.Node
    Dim szFolderType As String


    ' Decide which type of folder to Set the tree nodes icon

    Select Case ObjRoot.DefaultItemType
        Case olTaskItem:
            szFolderType = "task"
        Case olAppointmentItem:
            szFolderType = "calendar"
        Case olMailItem:
            szFolderType = "mail"
        Case olContactItem:
            szFolderType = "contact"
        Case olJournalItem:
            szFolderType = "journal"
        Case olNoteItem:
            szFolderType = "notes"
        Case Else
            szFolderType = "folder"
    End Select

    ' Generate an index For this root based On its parents code plus its
    ' peer level code
    szNodeIndex = szParentCode + Chr$(64 + n)

    ' Add a child node Or create the tree root depending On its parent
    If (szParentCode = "") Then
        Set nodX = vntTree.Nodes.Add(, , szNodeIndex, ObjRoot.Name, "outlook")
    Else
        Set nodX = vntTree.Nodes.Add(szParentCode, tvwChild, szNodeIndex, _
         ObjRoot.Name, szFolderType)
    End If

    nodX.EnsureVisible

    ' Iterate through all child folders of this folder And add them in turn
    For i = 1 To ObjRoot.Folders.Count
        AddLeaf ObjRoot.Folders(i), vntTree, szNodeIndex, i
    Next i

End Sub

Private Sub Cancel_Click()
'
'   Function:   Cancel_Click()
'
'   Parameters: none
'
'   Purpose:
'               Closes the FolderSelection form And undoes the chosen folder
'
'   Updates:    24-Jan-02

    lblPath.Caption = szUndoPath
    frmFolderSelection.Hide

End Sub
```

```
Private Sub OK_Click()
'
'   Function:   OK_Click()
'
'   Parameters: none
'
'   Purpose:
'               Closes the FolderSelection form
'
'   Updates:    24-Jan-02

    frmFolderSelection.Hide

End Sub

Private Sub OutlookFolderView_NodeClick(ByVal Node As MSComctlLib.Node)
'
'   Function:   OutlookFolderView_NodeClick()
'
'   Parameters: Node - unused
'
'   Purpose:
'               Update the Path shown at bottom of the dialog box when selection i
s changed
'
'   Updates:    24-Jan-02

    lblPath.Caption = OutlookFolderView.SelectedItem.FullPath

End Sub
```

3.4 frmImportReport

```
Private Sub OK_Click()
'
'    Function:    OK_Click()
'
'    Parameters: none
'
'    Purpose:     Closes the Report box
'
'    Updates:     24-Jan-02
     frmImportReport.Hide
End Sub
```

3.5 frmMain

```
Private Sub btnOptions_Click()
'   Function:   btnOptions_Click()
'
'   Parameters: none
'
'   Purpose:    Displays the Options Dialog box
'
'   Updates:    24-Jan-02

    frmOptions.Show vbModal
    Unload frmOptions
End Sub

Private Sub Form_Load()
'
'   Function:   Form_Load()
'
'   Parameters: none
'
'   Purpose:
'               Adds an "About" menu Option to the system menu And
'               retrieves the application user settings from the registry when
'               the application starts
'
'   Updates:     23-Jan-02

    gWH = Me.hwnd
    Dim hw As Long
    Dim hMenu As Long
    Dim X As Long
    'First we need to add our own menu item
    'to the System Menu
    hw = Me.hwnd
    hMenu = GetSystemMenu(hw, False)
    X = AppendMenu(hMenu, MF_SEPARATOR, 0, "")
    X = AppendMenu(hMenu, MF_STRING, SC_NEWMENU, "&About...")
    hook

    bCheckSubject = (QueryValue("Software\ContactImporter", "CheckSubjectLine") =
"Yes")
    szInboxFolder = QueryValue("Software\ContactImporter", "InboxFolder")
    szContactFolder = QueryValue("Software\ContactImporter", "ContactFolder")
    szSubjectLine = QueryValue("Software\ContactImporter", "SubjectLine")
    szDelimiter = QueryValue("Software\ContactImporter", "Delimiter")
    szContactUpdate = QueryValue("Software\ContactImporter", "ContactUpdate")
    imgDrivingDev_Click

End Sub

Private Sub Form_Unload(Cancel As Integer)
'
'   Function:   Form_UnLoad()
'
'   Parameters: none
'
'   Purpose:
'               Unhooks the "About" menu item from the system menu
'               And stores the application user settings in the registry
'
'
'   Updates:     23-Jan-02

    Unhook


    CreateNewKey "Software\ContactImporter\", HKEY_CURRENT_USER
    SetKeyValue "Software\ContactImporter", "InboxFolder", szInboxFolder, REG_SZ
    SetKeyValue "Software\ContactImporter", "ContactFolder", szContactFolder, REG_
SZ
    SetKeyValue "Software\ContactImporter", "SubjectLine", szSubjectLine, REG_SZ
    SetKeyValue "Software\ContactImporter", "Delimiter", szDelimiter, REG_SZ
    SetKeyValue "Software\ContactImporter", "ContactUpdate", szContactUpdate, REG_
SZ
```

```
    If (bCheckSubject = True) Then
        SetKeyValue "Software\ContactImporter", "CheckSubjectLine", "Yes", REG_SZ
    Else
        SetKeyValue "Software\ContactImporter", "CheckSubjectLine", "No", REG_SZ
    End If


End Sub


Private Sub imgDrivingDev_Click()
'
'   Function:   imgDrivingDev_Click()
'
'   Parameters: none
'
'   Purpose:
'               Displays the About box when the Image On the main form
'               Is clicked
'
'   Updates:     24-Jan-02

    frmAbout.Show vbModal
    Unload frmOptions

End Sub
```

3.6 frmOptions

```
Option Explicit

Private Sub btnBrowseInbox_Click()
'
'   Function:   btnBrowseInbox_Click()
'
'   Parameters: none
'
'   Purpose:
'               Sets the dialog box title And currently selected path For the
'               Inbox folder And brings up the folder selection dialog box
'               to allow user to change the selection
'
'   Updates:    23-Jan-02

    frmFolderSelection.Caption = "Select Inbox (Source) Folder"
    frmFolderSelection.lblPath.Caption = txtInboxFolder
    frmFolderSelection.Show vbModal
    txtInboxFolder = frmFolderSelection.lblPath.Caption
    Unload frmFolderSelection

End Sub

Private Sub btnBrowseContact_Click()
'
'   Function:   btnBrowseContact_Click()
'
'   Parameters: none
'
'   Purpose:
'               Sets the dialog box title And currently selected path For the
'               contact folder And brings up the folder selection dialog box
'               to allow user to change the selection
'
'   Updates:     23-Jan-02

    frmFolderSelection.Caption = "Select Contact (Destination) Folder"
    frmFolderSelection.lblPath.Caption = txtContactFolder
    frmFolderSelection.Show vbModal
    txtContactFolder = frmFolderSelection.lblPath.Caption
    Unload frmFolderSelection

End Sub


Private Sub SetCheck()
'
'   Function:   SetCheck()
'
'   Parameters: none
'
'   Purpose:
'               When users en/disables the email subject line checking change the
'               edit box so that user can/Not edit the subject line. Give colour
'               feedback On the edit box.
'
'   Updates:    23-Jan-02

    If (CheckSubject.Value = vbUnchecked) Then
        ' Disable subject line checking
        lblSubject.Enabled = False
        editSubject.Enabled = False
        editSubject.Locked = True
        editSubject.BackColor = &H8000000F
    Else
        ' Enable subject line checking
        lblSubject.Enabled = True
        editSubject.Enabled = True
        editSubject.Locked = False
        editSubject.BackColor = &H80000009
    End If

End Sub
```

```
Private Sub CheckSubject_Click()
'
'   Function:   CheckSubject_Click()
'
'   Parameters: none
'
'   Purpose:
'               Update status of controls when the subject text box Is clicked
'
'   Updates:    23-Jan-02

    SetCheck
End Sub

Private Sub Form_Load()
'
'   Function:   Form_Load()
'
'   Parameters: none
'
'   Purpose:
'               Initilises the Options form
'
'   Updates:     23-Jan-02

    If (bCheckSubject = True) Then
        CheckSubject.Value = vbChecked
    Else
        CheckSubject.Value = vbUnchecked
    End If

    If (szDelimiter = "[") Then
        optSquareBrackets.Value = True
    Else
        optQuotes = True
    End If

    txtContactUpdate.Text = szContactUpdate
    txtInboxFolder.Caption = szInboxFolder
    txtContactFolder.Caption = szContactFolder
    editSubject.Text = szSubjectLine

    SetCheck

End Sub


Private Sub OK_Click()
'
'   Function:   OK_Click()
'
'   Parameters: none
'
'   Purpose:
'               Accepts all the user settings from the Option dialog And close it
'
'   Updates:    23-Jan-02


    If (CheckSubject.Value = vbChecked) Then
        bCheckSubject = True
    Else
        bCheckSubject = False
    End If

    If (optSquareBrackets.Value = True) Then
        szDelimiter = "["
    Else
        szDelimiter = Chr$(34)
    End If

    szSubjectLine = editSubject.Text
    szContactUpdate = txtContactUpdate.Text
    szInboxFolder = txtInboxFolder.Caption
    szContactFolder = txtContactFolder.Caption
    frmOptions.Hide

End Sub
```

```
Private Sub Cancel_Click()
'
'   Function:   Cancel_Click()
'
'   Parameters: none
'
'   Purpose:
'               Hides the Options form when Cancel button Is clicked
'
'   Updates:    23-Jan-02

    frmOptions.Hide

End Sub
```

3.7 RegistryAccess

```
Option Explicit
'
'   Function:   Registry Access Functions
'
'   Parameters: n/a
'
'   Purpose:
'               Following code taken from MSDN to allow app. settings to be stored
 And
'               retrieved from the registry under the local user hive And Not unde
r VB
'               programs And settings.
'
'               (search For "HOWTO: Use the Registry API to Save And Retrieve Sett
ing")
'

   Public Const REG_SZ As Long = 1
   Public Const REG_DWORD As Long = 4

   Public Const HKEY_CLASSES_ROOT = &H80000000
   Public Const HKEY_CURRENT_USER = &H80000001
   Public Const HKEY_LOCAL_MACHINE = &H80000002
   Public Const HKEY_USERS = &H80000003

   Public Const ERROR_NONE = 0
   Public Const ERROR_BADDB = 1
   Public Const ERROR_BADKEY = 2
   Public Const ERROR_CANTOPEN = 3
   Public Const ERROR_CANTREAD = 4
   Public Const ERROR_CANTWRITE = 5
   Public Const ERROR_OUTOFMEMORY = 6
   Public Const ERROR_ARENA_TRASHED = 7
   Public Const ERROR_ACCESS_DENIED = 8
   Public Const ERROR_INVALID_PARAMETERS = 87
   Public Const ERROR_NO_MORE_ITEMS = 259

   Public Const KEY_QUERY_VALUE = &H1
   Public Const KEY_SET_VALUE = &H2
   Public Const KEY_ALL_ACCESS = &H3F

   Public Const REG_OPTION_NON_VOLATILE = 0

   Declare Function RegCloseKey Lib "advapi32.dll" _
   (ByVal hKey As Long) As Long
   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, ByVal lpSecurityAttributes _
   As Long, phkResult As Long, lpdwDisposition As Long) As Long
   Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias _
   "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, _
   ByVal ulOptions As Long, ByVal samDesired As Long, phkResult As _
   Long) As Long
   Declare Function RegQueryValueExString Lib "advapi32.dll" Alias _
   "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As _
   String, ByVal lpReserved As Long, lpType As Long, ByVal lpData _
   As String, lpcbData As Long) As Long
   Declare Function RegQueryValueExLong Lib "advapi32.dll" Alias _
   "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As _
   String, ByVal lpReserved As Long, lpType As Long, lpData As _
   Long, lpcbData As Long) As Long
   Declare Function RegQueryValueExNULL Lib "advapi32.dll" Alias _
   "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As _
   String, ByVal lpReserved As Long, lpType As Long, ByVal lpData _
   As Long, lpcbData As Long) As Long
   Declare Function RegSetValueExString Lib "advapi32.dll" Alias _
   "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName As String, _
   ByVal Reserved As Long, ByVal dwType As Long, ByVal lpValue As _
   String, ByVal cbData As Long) As Long
   Declare Function RegSetValueExLong Lib "advapi32.dll" Alias _
   "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName As String, _
   ByVal Reserved As Long, ByVal dwType As Long, lpValue As Long, _
   ByVal cbData As Long) As Long
```

```
Public Sub CreateNewKey(sNewKeyName As String, lPredefinedKey As Long)
       Dim hNewKey As Long         'handle to the New key
       Dim lRetVal As Long         'result of the RegCreateKeyEx Function

       lRetVal = RegCreateKeyEx(lPredefinedKey, sNewKeyName, 0&, _
                 vbNullString, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, _
                 0&, hNewKey, lRetVal)
       RegCloseKey (hNewKey)
   End Sub
   Public Sub SetKeyValue(sKeyName As String, sValueName As String, _
   vValueSetting As Variant, lValueType As Long)
       Dim lRetVal As Long         'result of the SetValueEx Function
       Dim hKey As Long         'handle of open key

       'open the specified key
       lRetVal = RegOpenKeyEx(HKEY_CURRENT_USER, sKeyName, 0, _
                               KEY_SET_VALUE, hKey)
       lRetVal = SetValueEx(hKey, sValueName, lValueType, vValueSetting)
       RegCloseKey (hKey)
   End Sub



   Public Function SetValueEx(ByVal hKey As Long, sValueName As String, _
   lType As Long, vValue As Variant) As Long
       Dim lValue As Long
       Dim sValue As String
       Select Case lType
           Case REG_SZ
               sValue = vValue & Chr$(0)
               SetValueEx = RegSetValueExString(hKey, sValueName, 0&, _
                                                lType, sValue, Len(sValue))
           Case REG_DWORD
               lValue = vValue
               SetValueEx = RegSetValueExLong(hKey, sValueName, 0&, _
   lType, lValue, 4)
           End Select
   End Function

   Function QueryValueEx(ByVal lhKey As Long, ByVal szValueName As _
   String, vValue As Variant) As Long
       Dim cch As Long
       Dim lrc As Long
       Dim lType As Long
       Dim lValue As Long
       Dim sValue As String

       On Error GoTo QueryValueExError

       ' Determine the size And type of data to be read
       lrc = RegQueryValueExNULL(lhKey, szValueName, 0&, lType, 0&, cch)
       If lrc <> ERROR_NONE Then Error 5

       Select Case lType
           ' For strings
           Case REG_SZ:
               sValue = String(cch, 0)

   lrc = RegQueryValueExString(lhKey, szValueName, 0&, lType, _
   sValue, cch)
               If lrc = ERROR_NONE Then
                   vValue = Left$(sValue, cch - 1)
               Else
                   vValue = Empty
               End If
           ' For DWORDS
           Case REG_DWORD:
   lrc = RegQueryValueExLong(lhKey, szValueName, 0&, lType, _
   lValue, cch)
               If lrc = ERROR_NONE Then vValue = lValue
           Case Else
               'all other data types Not supported
               lrc = -1
       End Select

QueryValueExExit:
       QueryValueEx = lrc
       Exit Function
```

```
QueryValueExError:
        Resume QueryValueExExit
    End Function


Public Function QueryValue(sKeyName As String, sValueName As String) As Variant
        Dim lRetVal As Long         'result of the API functions
        Dim hKey As Long          'handle of opened key
        Dim vValue As Variant       'setting of queried value

        lRetVal = RegOpenKeyEx(HKEY_CURRENT_USER, sKeyName, 0, _
    KEY_QUERY_VALUE, hKey)
        lRetVal = QueryValueEx(hKey, sValueName, vValue)

        RegCloseKey (hKey)
        QueryValue = vValue
    End Function
```

3.8 SystemMenu

```
Option Explicit
'
'   Function:   Custom Menu Functions
'
'   Parameters: n/a
'
'   Purpose:
'               Following code taken from MSDN to allow an "About" menu Option to
be added
'               to the system menu As this application Is a dialog based app.
'
'               (search For "Intercepting Windows Messages in Visual Basic")
'
Declare Function CallWindowProc Lib "user32" Alias _
   "CallWindowProcA" (ByVal lpPrevWndFunc As Long, _
   ByVal hwnd As Long, ByVal Msg As Long, ByVal _
   wParam As Long, ByVal lParam As Long) As Long
Declare Function SetWindowLong Lib "user32" Alias _
   "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex _
   As Long, ByVal dwNewLong As Long) As Long
Declare Function AppendMenu Lib "user32" Alias _
   "AppendMenuA" (ByVal hMenu As Long, ByVal wFlags _
   As Long, ByVal wIDNewItem As Long, ByVal _
   lpNewItem As String) As Long
Declare Function GetSystemMenu Lib "user32" (ByVal _
   hwnd As Long, ByVal bRevert As Long) As Long
Public Const WM_SYSCOMMAND = &H112
Public Const MF_STRING = &H0
Public Const MF_SEPARATOR = &H800
Public Const SC_NEWMENU = 1
Public Const GWL_WNDPROC = -4
Global lpPrevWndProc As Long
Global gWH As Long

Public Sub hook()
   lpPrevWndProc = SetWindowLong(gWH, GWL_WNDPROC, _
      AddressOf WindowProc)
End Sub
Public Sub Unhook()
   Dim temp As Long
   temp = SetWindowLong(gWH, GWL_WNDPROC, lpPrevWndProc)
End Sub
Function WindowProc(ByVal hw As Long, ByVal uMsg As _
   Long, ByVal wParam As Long, ByVal lParam As Long) _
   As Long
  'We need to trap the WM_SYSCOMMAND message to
  'determine when the user has clicked On our
  'New menu item. The message Is stored in uMsg.
  'Our New menu item Is identified As SC_NEWMENU
   If uMsg = WM_SYSCOMMAND Then
   Select Case wParam And &HFFFF&
      Case SC_NEWMENU
        frmAbout.Show vbModal
        Unload frmAbout
   End Select
   End If
   'Always call the original handler when done
   WindowProc = CallWindowProc(lpPrevWndProc, hw, _
      uMsg, wParam, lParam)
End Function
```