

```

00001 Option Explicit
00002 ' demo project showing how to create a custom MSDE install
00003 ' by Bryan Stafford of New Vision Software - newvision@mvps.org
00004 ' this demo is released into the public domain "as is" without
00005 ' warranty or guaranty of any kind. In other words, use at
00006 ' your own risk.
00007 '
00008 ' Copyright © 2002 Bryan Stafford/New Vision Software
00009 '
00010 ' the code in this demo is copyrighted and may only be used in
00011 ' accordance with the rules set forth in the "House Rules"
00012 ' section of the web page found at http://www.mvps.org/vbvision/
00013 '
00014 ' see the general declararions section of MMain.bas for more info
00015 ' on this project
00016
00017 Private m_sServerName As String
00018 Private m_sUserID As String
00019 Private m_sPassword As String
00020 Private m_bNTAuthentication As Boolean
00021
00022 Private m_rc As RECT
00023 Private mRetVal As NAVIGATE_FLAGS

```

```

00025
00026 > Public Function DisplayDialog(oPicture As StdPicture, ByVal IpRect&, ByVal sServerName$, ByVal sUserID$,
    > ByVal sPassword$, ByVal bNTAuthentication As Boolean) As NAVIGATE_FLAGS
00027
00028 ' set the wait cursor in case loading the form takes a while
00029 Screen.MousePointer = vbHourglass
00030
00031 Dim hSysMenu&
00032
00033 ' first thing to do is get the handle to the system menu for this form
00034 hSysMenu = GetSystemMenu(hWnd, False)
00035
00036 Call DeleteMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
00037
00038
00039 If oPicture Is Nothing Then Set oPicture = CreateTitlePicture(hWnd)
00040
00041 Set Picture = oPicture
00042
00043 CopyMemory m_rc, ByVal IpRect, Len(m_rc)
00044
00045 With m_rc
00046 Call SetWindowPos(hWnd, 0&, .Left, .Top, 0&, 0&, SWP_NOSIZE Or SWP_NOZORDER)
00047 End With
00048
00049 m_sServerName = sServerName
00050 m_sUserID = sUserID
00051 m_sPassword = sPassword
00052 m_bNTAuthentication = bNTAuthentication
00053
00054 Screen.MousePointer = vbDefault
00055
00056 tmrConfigureServer.Enabled = True
00057
00058 Show vbModal, MMain.TaskbarProxy
00059
00060
00061 CopyMemory ByVal IpRect, m_rc, Len(m_rc)
00062
00063 DisplayDialog = mRetVal
00064
00065 End Function

```

```

00066
00067 Private Sub cmdNav_Click(Index As Integer)
00068
00069 Select Case Index
00070 Case eHelp
00071
00072 Case eCancel
00073 If AskAboutCancel() Then
00074 mRetVal = Index
00075
00076 Unload Me
00077 End If

```

```

00078 1 2
00079 } Case Else
00080     mRetVal = Index
00081 }
00082     Unload Me
00083 } End Select
00084 }
00085 } End Sub

00086
00087 } Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
00088     If mRetVal = 0 Then Cancel = True
00089 } End Sub

00090
00091 } Private Sub Form_Unload(Cancel As Integer)
00092     Call GetWindowRect(hWnd, m_rc)
00093 }
00094 } End Sub
00095 }

00096
00097 } Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
00098     ' keep the user from using Alt+F4 to close the form. notice
00099     ' that we have the KeyPreview property for the form set to True
00100     If KeyCode = vbKeyF4 Then KeyCode = 0
00101 } End Sub

00102
00103 } Private Sub tmrConfigureServer_Timer()
00104     Dim sMsg$
00105     tmrConfigureServer.Enabled = False
00106
00107     If ConfigureSQLServer(m_sServerName, m_sUserID, m_sPassword, m_bNTAuthentication, Me) Then
00108         lblConfigInfo.Caption = "The SQL Server desktop edition has been successfully configured." & vbNewLine
00109         & vbNewLine & "Click ""Finish"" to exit."
00110     }
00111     cmdNav(eFinish).Enabled = True
00112 } Else
00113     sMsg = "An error ocured while configuring the MSDE installation."
00114     MsgBox sMsg, vbInformation, vMSGBOXCAPTION
00115     lblConfigInfo.Caption = "Errors occurred durring the configurati on process." & vbNewLine & vbNewLine &
00116     "Unable to conti nue wi th server confi guration."
00117 } End If
00118     cmdNav(eCancel).Enabled = True
00119 } End Sub
00120 }

00121
00122 } Public Sub ConfigurationCallback(ByVal sInfoText$)
00123     lblConfigInfo.Caption = sInfoText
00124     Refresh
00125 } End Sub
00126 }
00127 }
00128 }
00129 }
00130 }
00131 }
00132 }
00133 }
00134 }
00135 }
00136 }
00137 }
00138 }
00139 }
00140 }

```

```

00141 Option Explicit
00142 ' demo project showing how to create a custom MSDE install
00143 ' by Bryan Stafford of New Vision Software - newvision@mmps.org
00144 ' this demo is released into the public domain "as is" without
00145 ' warranty or guaranty of any kind. In other words, use at
00146 ' your own risk.
00147 '
00148 ' Copyright © 2002 Bryan Stafford/New Vision Software
00149 '
00150 ' the code in this demo is copyrighted and may only be used in
00151 ' accordance with the rules set forth in the "House Rules"
00152 ' section of the web page found at http://www.mmps.org/vbvision/
00153 '
00154 ' see the general declararions section of MMain.bas for more info
00155 ' on this project
00156
00157
00158 Private Enum EDIT_INDEXES
00159     eSQLServerName = 0&
00160     eLogi nID = 1&
00161     eLogi nPassword = 2&
00162 End Enum
00163
00164 Private m_sServerName As String
00165 Private m_sUserID As String
00166 Private m_sPassword As String
00167 Private m_bNTAuthnenti cation As Boolean
00168
00169
00170 Private m_rc As RECT
00171 Private m_RetVal As NAVIGATE_FLAGS
00172
00173 Public Function DisplayDialog(oPicture As StdPicture, ByVal IpRect&, ByRef sServerName$, ByRef sUserID$,
» ByRef sPassword$, ByRef bNTAuthnenti cation As Boolean) As NAVIGATE_FLAGS
00174     ' set the wait cursor in case loading the form takes a while
00175     Screen.MousePoi nter = vbHourglass
00176
00177     Dim hSysMenu&
00178
00179     ' first thing to do is get the handle to the system menu for this form
00180     hSysMenu = GetSystemMenu(hWnd, False)
00181
00182     Call DeleteMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
00183
00184
00185     If oPicture Is Nothing Then Set oPicture = CreateTi tlePicture(hWnd)
00186
00187     Set Picture = oPicture
00188
00189     CopyMemory m_rc, ByVal IpRect, Len(m_rc)
00190
00191
00192     With m_rc
00193     Call SetWi ndowPos(hWnd, 0&, .Left, .Top, 0&, 0&, SWP_NOSI ZE Or SWP_NOZORDER)
00194     End Wi th
00195
00196     ckNTSecuri ty.Caption = "Use Windows" & vbNewLi ne & "NT Authenti cation"
00197
00198
00199     ' can't use NT auth on Win9x
00200     If GetOSVersi on() < [eNT4.0] Then ckNTSecuri ty.Enabled = False
00201
00202
00203     lblInstructions.Caption = "Please enter the desi red server i nformation or accept the default values
»     below"
00204     txServerI nfo(eSQLServerName).MaxLength = 16
00205
00206
00207     txServerI nfo(eSQLServerName).Text = sServerName
00208     txServerI nfo(eLogi nID).Text = sUserID
00209     txServerI nfo(eLogi nPassword).Text = sPassword
00210
00211     If bNTAuthenti cation Then ckNTSecuri ty.Value = vbChecked
00212
00213
00214     Screen.MousePoi nter = vbDefaul t
00215
00216     Show vbModal, MMai n.TaskbarProxy
00217 1

```

```

1
00218
00219     sServerName = m_sServerName
00220     sUserID = m_sUserID
00221     sPassword = m_sPassword
00222
00223     bNTAuthentication = m_bNTAuthentication
00224
00225
00226     CopyMemory ByVal lpRect, m_rc, Len(m_rc)
00227
00228     DisplayDialog = m_ReturnVal
00229
00230 End Function
-----
00231
00232 Private Sub cmdNav_Click(Index As Integer)
00233
00234     Dim sMsg$, bNotEnoughData As Boolean
00235
00236
00237     Select Case Index
00238     Case eHelp
00239
00240     Case eCancel
00241         If AskAboutCancel() Then
00242             m_ReturnVal = Index
00243
00244             Unload Me
00245         End If
00246
00247     Case eNext
00248         Dim oRegistry As CRegistry
00249         Set oRegistry = New CRegistry
00250
00251         With oRegistry
00252             .Key = HKEY_LOCAL_MACHINE
00253
00254             If Len(.ReadOnlyReadSetting("SOFTWARE\Microsoft\Microsoft SQL Server\" & txServerInfo(0).Text,
00255 »         "SQLPath", vbNullString)) <> 0 Then
00256 »             sMsg = "The specified instance name is already in use on this system. Cannot install a new
00257 »             instance using the provided server name."
00258
00259             bNotEnoughData = True
00260         End If
00261     End With
00262
00263     Set oRegistry = Nothing
00264
00265     If bNotEnoughData = False Then
00266         If ckNTSecurity.Value = vbChecked Then
00267             If Len(txServerInfo(eSQLServerName).Text) = 0 Then
00268                 sMsg = "You must enter a SQL Server name before you may continue."
00269
00270                 bNotEnoughData = True
00271             End If
00272         Else
00273             If (Len(txServerInfo(eSQLServerName).Text) = 0) Or (Len(txServerInfo(eLoginID).Text) = 0) Then
00274                 sMsg = "You must enter a SQL Server name, User ID and Password before you may continue."
00275
00276                 bNotEnoughData = True
00277             ElseIf (Len(txServerInfo(eLoginPassword).Text) = 0) Then
00278 »                 If MsgBox("No value was entered for the password." & vbNewLine & vbNewLine & "Are you sure you
00279 »                 want to continue with a blank password?", _
00280 »                 vbYesNo Or vbQuestion, nvMSGBOXCAPTION) = vbNo Then
00281                     bNotEnoughData = True
00282                 End If
00283             End If
00284         End If
00285     End If
00286
00287     If bNotEnoughData Then
00288         MsgBox sMsg, vbInformation, nvMSGBOXCAPTION
00289     Else
00290         m_sServerName = txServerInfo(eSQLServerName).Text
00291         m_sUserID = txServerInfo(eLoginID).Text
00292         m_sPassword = txServerInfo(eLoginPassword).Text
00293
1 2 3

```

```

00294 1 2 3 m_bNTAutnenti cation = (ckNTSecuri ty.Val ue = vbChecked)
00295     m_RetVal = Index
00296     Unl oad Me
00297     End If
00298
00299
00300
00301
00302 } Case eBack
00303     m_sServerName = txServerI nfo(eSQLServerName). Text
00304     m_sUserID = txServerI nfo(eLogi nID). Text
00305     m_sPassword = txServerI nfo(eLogi nPassword). Text
00306     m_bNTAutnenti cation = (ckNTSecuri ty.Val ue = vbChecked)
00307     m_RetVal = Index
00308     Unl oad Me
00309     End Select
00310
00311
00312
00313
00314 } End Sub
00315
00316
00317
00318 } Private Sub Form_QueryUnl oad(Cancel As Integer, Unl oadMode As Integer)
00319     If m_RetVal = 0 Then Cancel = True
00320 } End Sub
00321
00322 } Private Sub Form_Unl oad(Cancel As Integer)
00323     Call GetWi ndowRect(hWnd, m_rc)
00324 } End Sub
00325
00326
00327
00328 } Private Sub Form_KeyDown(KeyCode As Integer, Shi ft As Integer)
00329     ' keep the user from using Alt+F4 to close the form. notice
00330     ' that we have the KeyPreview property for the form set to True
00331     If KeyCode = vbKeyF4 Then KeyCode = 0
00332 } End Sub
00333
00334 } Private Sub ckNTSecuri ty_Click()
00335     If ckNTSecuri ty.Val ue = vbChecked Then
00336         frSQLServi DPassword.Enabled = False
00337         I bl SQLI D(0).Enabled = False
00338         I bl SQLI D(1).Enabled = False
00339         txServerI nfo(eLogi nID).Enabled = False
00340         txServerI nfo(eLogi nPassword).Enabled = False
00341     Else
00342         frSQLServi DPassword.Enabled = True
00343         I bl SQLI D(0).Enabled = True
00344         I bl SQLI D(1).Enabled = True
00345         txServerI nfo(eLogi nID).Enabled = True
00346         txServerI nfo(eLogi nPassword).Enabled = True
00347     End If
00348 } End Sub
00349
00350
00351
00352
00353
00354
00355

```

```

00356 Option Explicit
00357 ' demo project showing how to create a custom MSDE install
00358 ' by Bryan Stafford of New Vision Software - newvision@mvp.org
00359 ' this demo is released into the public domain "as is" without
00360 ' warranty or guaranty of any kind. In other words, use at
00361 ' your own risk.
00362 '
00363 ' Copyright © 2002 Bryan Stafford/New Vision Software
00364 '
00365 ' the code in this demo is copyrighted and may only be used in
00366 ' accordance with the rules set forth in the "House Rules"
00367 ' section of the web page found at http://www.mvps.org/vbvision/
00368 '
00369 ' see the general declararions section of MMain.bas for more info
00370 ' on this project
00371
00372
00373 Private m_sServerName As String
00374 Private m_sUserID As String
00375 Private m_sPassword As String
00376 Private m_bNTAuthentication As Boolean
00377
00378 Private m_rc As RECT
00379 Private mRetVal As NAVIGATE_FLAGS

```

```

00380
00381 Public Function DisplayDialog(oPicture As StdPicture, ByVal IpRect&, ByVal sServerName$, ByVal sUserID$,
ByVal sPassword$, ByVal bNTAuthentication As Boolean) As NAVIGATE_FLAGS
>
00382     ' set the wait cursor in case loading the form takes a while
00383     Screen.MousePointer = vbHourglass
00384
00385     Dim hSysMenu&
00386
00387     ' first thing to do is get the handle to the system menu for this form
00388     hSysMenu = GetSystemMenu(hWnd, False)
00389
00390     Call DeleteMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
00391
00392
00393
00394     Set Picture = oPicture
00395
00396     CopyMemory m_rc, ByVal IpRect, Len(m_rc)
00397
00398     With m_rc
00399     Call SetWindowPos(hWnd, 0&, .Left, .Top, 0&, 0&, SWP_NOSIZE Or SWP_NOZORDER)
00400     End With
00401
00402     m_sServerName = sServerName
00403     m_sUserID = sUserID
00404     m_sPassword = sPassword
00405     m_bNTAuthentication = bNTAuthentication
00406
00407     lblInstallMessage.Caption = "Please be patient while SQL Server is installed on your system." & vbNewLine
>     & vbNewLine _
00408     & "Portions of this process may take several minutes" & vbNewLine & "and it may appear as if the
>     application has stalled..."
00409
00410
00411
00412     Screen.MousePointer = vbDefault
00413
00414     tmrStartInstall.Enabled = True
00415
00416     Show vbModal, MMain.TaskbarProxy
00417
00418
00419     CopyMemory ByVal IpRect, m_rc, Len(m_rc)
00420
00421
00422     DisplayDialog = mRetVal
00423
00424 End Function

```

```

00425
00426 Private Sub cmdNav_Click(Index As Integer)
00427
00428     Select Case Index
00429     Case eHelp
00430
1 2

```

```

1 2
00431 } Case eCancel
00432   | If AskAboutCancel () Then
00433   |   | mRetVal = Index
00434   |
00435   |   | Unload Me
00436   |   | End If
00437   |
00438   | Case Else
00439   |   | mRetVal = Index
00440   |   |
00441   |   | Unload Me
00442   |   | End Select
00443   |
00444 } End Sub

00445
00446 } Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
00447   | If mRetVal = 0 Then Cancel = True
00448 } End Sub

00449
00450 } Private Sub Form_Unload(Cancel As Integer)
00451   |
00452   | Call GetWindowRect(hWnd, m_rc)
00453   |
00454 } End Sub

00455
00456 } Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
00457   | ' keep the user from using Alt+F4 to close the form. notice
00458   | ' that we have the KeyPreview property for the form set to True
00459   | If KeyCode = vbKeyF4 Then KeyCode = 0
00460 } End Sub

00461
00462 } Private Function GetAvailableMSIPackageName() As String
00463   |
00464   | ' - If you already have MSDE 2000 installed you can check the following registry key to find which .msi
00465   | ' file the setup used for the specific MSDE 2000 instance:
00466   | '
00467   | '     Look for the existing MSDE 2000 instance name in the following registry key:
00468   | '
00469   | '     HKEY_CLASSES_ROOT\Installer\Products\ID_Number
00470   | '
00471   | '     The ProductName value displays the instance name. For example, "ProductName" = Microsoft SQL
00472   | '     Server Desktop Engine (MYINSTANCE)
00473   | '     Locate the following registry subkey:
00474   | '
00475   | '     HKEY_CLASSES_ROOT\Installer\Products\ID_Number\SourceList
00476   | '
00477   | '     The PackageName key value shows the .msi file. For example, "PackageName"="SqlRun01.msi"
00478   | '
00479   | ' - If MSDE 2000 is already installed, check the MSDE 2000 corresponding instance registry key to find
00480   | ' the ProductCode for the MSDE instance. Next, use the ProductCode
00481   | ' value to match up the corresponding package file from the table that follows. Please note that the
00482   | ' following table information only applies to package files provided
00483   | ' with the Microsoft SQL Server Desktop Engine Setup CD and certain other Microsoft products that
00484   | ' include a custom install. The table is not exhaustive and does not
00485   | ' apply to any third-party developed setup packages.
00486   | '
00487   | '     Default Instance
00488   | '
00489   | '     If the MSDE instance is a default instance, check the following registry key for the ProductCode
00490   | ' value:
00491   | '
00492   | '     HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer\Setup
00493   | '
00494   | '     Named Instance
00495   | '
00496   | '     If the MSDE instance is a named instance, check the following registry key for the ProductCode
00497   | ' value:
00498   | '
00499   | '     HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\INSTANCENAME\Setup
00500   | '
00501   | '     {E09B48B5-E141-427A-AB0C-D3605127224A} --> SqlRun01.msi
00502   | '     {689404D2-1C94-44B3-9203-BEC5594FDA7A} --> SqlRun02.msi
00503   | '     {EFB70B01-B1F3-4960-AB69-4A280084A60C} --> SqlRun03.msi

```

```

00501 1 ' {C2736CA7-76E1-4DOC-B590-483A7FFD18DA} --> Sql Run04. msi
00502 ' {FE7E950B-220A-4182-B5CA-19397244DCFD} --> Sql Run05. msi
00503 ' {7E5C338B-E77E-4CB4-9C1D-FB67B56B3B19} --> Sql Run06. msi
00504 ' {F07E35BF-8B03-4777-9B5E-AE90E4FF0932} --> Sql Run07. msi
00505 ' {C5B59406-E985-4187-84E8-68E2D9F89A47} --> Sql Run08. msi
00506 ' {D7CE240C-0F3B-4C40-9278-COB90E533652} --> Sql Run09. msi
00507 ' {A519AE9C-7C79-4C5B-9127-8F46D648D5A4} --> Sql Run10. msi
00508 ' {4541DA32-2108-43E9-9915-C71B9DE77048} --> Sql Run11. msi
00509 ' {A5C1C914-4EF7-40ED-9BCE-FCEB4BB0C19D} --> Sql Run12. msi
00510 ' {9FCE5BBD-D85F-4905-8A0C-12A3A86C2434} --> Sql Run13. msi
00511 ' {F4E46404-2578-4955-B748-547957F08AB1} --> Sql Run14. msi
00512 ' {B7300824-E68F-45F1-BAC1-5F15636C346F} --> Sql Run15. msi
00513 ' {CD59EA85-6CBF-4C08-BE59-6C628B3D8F54} --> Sql Run16. msi
00514
00515
00516 Dim hKey&, lRet&, nCurIdx&, sKeyName$, sProductCode$, sClassName$, nKeyLen&, nClassNameLen&, lpFT As FILETIME

00517
00518 Dim i&, sMSIFileName$, asMSIInstanceFiles(15) As String, oRegistry As CRegistry
00519 For i = 0 To 15
00520     asMSIInstanceFiles(i) = "Sql Run" & Format$(i + 1, "00") & ".msi"
00521 Next
00522
00523
00524 Set oRegistry = New CRegistry
00525
00526 With oRegistry
00527     .Key = HKEY_LOCAL_MACHINE
00528     sProductCode = .ReadOnlyReadSetting("SOFTWARE\Microsof\Microsoft SQL Server\Setup", "ProductCode", vbNullString
00529 )
00530 End With
00531
00532 If Len(sProductCode) Then Call RemoveMSIFileNameFromArray(GetMSIFromProductCode(sProductCode),
00533     asMSIInstanceFiles())
00534
00535
00536 ' check we can open key
00537 lRet = RegOpenKeyEx(HKEY_LOCAL_MACHINE, ByVal "SOFTWARE\Microsof\Microsof SQL Server", ByVal 0&,
00538     KEY_ENUMERATE_SUB_KEYS, hKey)
00539
00540 If lRet = 0 Then
00541     Do
00542         nKeyLen = MAX_PATH
00543         nClassNameLen = MAX_PATH
00544         sKeyName = String$(nKeyLen, vbNullChar)
00545         sClassName = String$(nClassNameLen, vbNullChar)
00546         lRet = RegEnumKeyEx(hKey, nCurIdx, sKeyName, nKeyLen, 0&, sClassName, nClassNameLen, lpFT)
00547         sKeyName = StripNulls(sKeyName)
00548         If lRet = ERROR_SUCCESS Then
00549             'RaiseEvent EnumValue(sKeyName, nCurIdx)
00550             sProductCode = oRegistry.ReadOnlyReadSetting("SOFTWARE\Microsof\Microsof SQL Server\" & sKeyName
00551                 & "\Setup", "ProductCode", vbNullString)
00552             If Len(sProductCode) Then Call RemoveMSIFileNameFromArray(GetMSIFromProductCode(sProductCode),
00553                 asMSIInstanceFiles())
00554         End If
00555         nCurIdx = nCurIdx + 1
00556     Loop While lRet = ERROR_SUCCESS
00557
00558 Call RegCloseKey(hKey)
00559 End If
00560
00561 Set oRegistry = Nothing
00562
00563 For i = 0 To 15
00564     If Len(asMSIInstanceFiles(i)) Then
00565         GetAvailablMSIPackageName = asMSIInstanceFiles(i)
00566     Exit For
00567 End If
00568
00569
00570
00571
00572
00573 2

```

```

00574 1 2 Next
00575
00576 End Function

```

```

00577
00578 Private Sub RemoveMSIFileNameFromArray(ByVal sMSIFileName$, asMSIFileNames() As String)
00579
00580     Dim i&
00581
00582     For i = 0 To UBound(asMSIFileNames)
00583         If LCase$(asMSIFileNames(i)) = LCase$(sMSIFileName) Then
00584             asMSIFileNames(i) = vbNullString
00585
00586             Exit For
00587         End If
00588     Next
00589
00590 End Sub

```

```

00591
00592 Private Function GetMSIFromProductCode(ByVal sProductCode$) As String
00593
00594     Select Case sProductCode
00595     Case "{E09B48B5-E141-427A-AB0C-D3605127224A}": GetMSIFromProductCode = "Sql Run01.msi"
00596     Case "{689404D2-1C94-44B3-9203-BEC5594FDA7A}": GetMSIFromProductCode = "Sql Run02.msi"
00597     Case "{EFB70B01-B1F3-4960-AB69-4A280084A60C}": GetMSIFromProductCode = "Sql Run03.msi"
00598     Case "{C2736CA7-76E1-4DOC-B590-483A7FFD18DA}": GetMSIFromProductCode = "Sql Run04.msi"
00599     Case "{FE7E950B-220A-4182-B5CA-19397244DCFD}": GetMSIFromProductCode = "Sql Run05.msi"
00600     Case "{7E5C338B-E77E-4CB4-9C1D-FB67B56B3B19}": GetMSIFromProductCode = "Sql Run06.msi"
00601     Case "{F07E35BF-8B03-4777-9B5E-AE90E4FF0932}": GetMSIFromProductCode = "Sql Run07.msi"
00602     Case "{C5B59406-E985-4187-84E8-68E2D9F89A47}": GetMSIFromProductCode = "Sql Run08.msi"
00603     Case "{D7CE240C-0F3B-4C40-9278-C0B90E533652}": GetMSIFromProductCode = "Sql Run09.msi"
00604     Case "{A519AE9C-7C79-4C5B-9127-8F46D648D5A4}": GetMSIFromProductCode = "Sql Run10.msi"
00605     Case "{4541DA32-2108-43E9-9915-C71B9DE77048}": GetMSIFromProductCode = "Sql Run11.msi"
00606     Case "{A5C1C914-4EF7-40ED-9BCE-FCEB4BBOC19D}": GetMSIFromProductCode = "Sql Run12.msi"
00607     Case "{9FCE5BBB-D85F-4905-8A0C-12A3A86C2434}": GetMSIFromProductCode = "Sql Run13.msi"
00608     Case "{F4E46404-2578-4955-B748-547957F08AB1}": GetMSIFromProductCode = "Sql Run14.msi"
00609     Case "{B7300824-E68F-45F1-BAC1-5F15636C346F}": GetMSIFromProductCode = "Sql Run15.msi"
00610     Case "{CD59EA85-6CBF-4C08-BE59-6C628B3D8F54}": GetMSIFromProductCode = "Sql Run16.msi"
00611     End Select
00612
00613 End Function

```

```

00614
00615 Private Sub tmrStartInstall_Timer()
00616
00617     Dim sMSIPath$, sCmdLine$, sRegKey$, nWaitState&, nExitCode&, nErrorCode&, sMsg$, udtStartInf As
    » STARTUPINFO, udtProclnf As PROCESS_INFORMATION
00618     Dim sInstallInstanceFile$, sTempDir$, sIniPath$, sLogFileMsg$, bDialogFound As Boolean
00619
00620     tmrStartInstall.Enabled = False
00621
00622     udtStartInf.cb = Len(udtStartInf)
00623
00624     sMSIPath = String$(MAX_PATH, 0)
00625
00626     If GetSystemDirectory(sMSIPath, MAX_PATH) Then
00627         sInstallInstanceFile = GetAvailableMSIPackageName()
00628
00629         If Len(sInstallInstanceFile) Then
00630             sMSIPath = StripNulls(sMSIPath)
00631             sMSIPath = ReturnShortPathName(sMSIPath)
00632
00633             If Len(sMSIPath) Then sMSIPath = AddBackslashToPath(sMSIPath)
00634
00635             sMSIPath = sMSIPath & "msiexec.exe"
00636
00637             If FileExists(sMSIPath) Then
00638                 sCmdLine = " /i " & sInstallInstanceFile
00639
00640                 Dim i&, sAllDrives$, sDrive$
00641
00642                 sAllDrives = String$(MAX_PATH, 0)
00643
00644                 Call GetLogicalDriveStrings(MAX_PATH, sAllDrives)
00645
00646                 For i = 1 To (MAX_PATH - 4) Step 4
00647                     sDrive = Mid$(sAllDrives, i, 3)
00648                 Next i
00649
00650

```

```

00651 1 2 3 4 5 If GetDriveType(sDrive) = DRIVE_FIXED Then
00652     sTempDir = sDrive
00653
00654     Exit For
00655 End If
00656 Next
00657
00658 If Len(sTempDir) = 0 Then sTempDir = "C:\\"
00659
00660 sTempDir = AddBackslashToPath(sTempDir)
00661
00662 If UCase$(m_sServerName) <> "DEFAULT" Then
00663     sCmdLine = sCmdLine & " INSTANCENAME=" & m_sServerName
00664 End If
00665
00666 sCmdLine = sCmdLine & " SAPWD=" & m_sPassword & " CALLBACK=nmvMSI_ERR!MSICBK"
00667
00668 If m_bNTAuthentication = False Then
00669     sCmdLine = sCmdLine & " SECURITYMODE=SQL"
00670 End If
00671
00672
00673 ' if you want to enable network protocols in MSDE 2k SP3a, you will need to
00674 ' include the following flag in your command line....
00675 '
00676 '     DISABLENETWORKPROTOCOLS=0
00677 '
00678
00679 sCmdLine = sCmdLine & " /qb"
00680
00681
00682
00683 sCmdLine = sCmdLine & " /I*v " & sTempDir & "msde_inst.log"
00684
00685
00686
00687 Dim sMSIErrPath$, sTempPath$, hMSIErrLib$, bDeleteMSIErrFile As Boolean
00688
00689 sTempPath = String$(MAX_PATH, 0)
00690
00691 If GetTempPath(MAX_PATH, sTempPath) Then
00692     sTempPath = AddBackslashToPath(StripNulls(sTempPath)) & "nmvMSI_ERR.dll"
00693
00694     If FileExists(sTempPath) = False Then bDeleteMSIErrFile = True
00695
00696     sMSIErrPath = AddBackslashToPath(App.Path) & "nmvMSI_ERR.dll"
00697
00698     If CopyFile(sMSIErrPath, sTempPath, API_FALSE) Then
00699         hMSIErrLib = LoadLibrary(sTempPath)
00700     End If
00701 End If
00702
00703
00704 Call InitializeMSILib 'nmvMSI_ERR.dll
00705
00706
00707 » If CreateProcess(sMSIPath, sCmdLine, 0&, 0&, API_FALSE, NORMAL_PRIORITY_CLASS, ByVal 0&, App.Path,
00708     udtStartInf, udtProcInf) Then
00709     Do
00710         If bDialogFound = False Then bDialogFound = DialogResult.CancelButton(udtProcInf.dwProcessId)
00711         Yield
00712         nWaitState = WaitForSingleObject(udtProcInf.hProcess, 500&)
00713     Loop Until (nWaitState = WAIT_OBJECT_0) Or (nWaitState = WAIT_FAILED)
00714
00715     Call GetExitCodeProcess(udtProcInf.hProcess, nExitCode)
00716
00717     Call CloseHandle(udtProcInf.hThread)
00718     Call CloseHandle(udtProcInf.hProcess)
00719
00720     nErrorCode = RetrieveMSIExitCode()
00721
00722     If hMSIErrLib Then
00723         Call FreeLibrary(hMSIErrLib)
00724         If bDeleteMSIErrFile Then Call DeleteFile(sTempPath)
00725         hMSIErrLib = 0
00726     End If
00727
00728     If (nExitCode = 0) And (nErrorCode <> 0) Then nExitCode = nErrorCode

```

```

00729 1 2 3 4 5
00730
00731 } Select Case nExitCode
00732 } Case ERROR_SUCCESS
00733     lblInstalMessage.Caption = "The SQL Server desktop edition has been successfully installed." &
00734     vbNewLine & vbNewLine & "Click ""Next"" to continue."
00735
00736     cmdNav(eNext).Enabled = True
00737
00738 } Case ERROR_SUCCESS_REBOOT_REQUIRED, ERROR_SUCCESS_REBOOT_INITIATED
00739     sRegKey = CStr(CDbl(Now))
00740
00741     Call SetRunKey("/Config", sRegKey)
00742     Call SetCommandLineKey(sRegKey, m_sServerName & "|" & m_sUserID & "|" & m_sPassword & "|" &
00743     CStr(m_bNTAuthentication))
00744
00745     If nExitCode = ERROR_SUCCESS_REBOOT_REQUIRED Then
00746         sMsg = "The system must be rebooted before the PAS Database installer can continue with the
00747         configuration of the MSDE install." & vbNewLine & vbNewLine &
00748         & "Choose ""Yes"" to reboot the system now or ""No"" to reboot later."
00749
00750         If MsgBox(sMsg, vbQuestion Or vbYesNo, nvMSGBOXCAPTION) = vbNo Then
00751             sMsg = "To automatically configure the database server, you must re-start the system with
00752             the PAS Database installer CD in the CD-ROM drive." &
00753             & " If the PAS Database installer application is not available when the system
00754             re-starts, you will need to run the " &
00755             & "PAS Database installer application using the ""/Config"" command line switch to
00756             configure the database server." &
00757             & " For more information on the ""/Config"" switch, see the PAS documentation." &
00758             vbNewLine & vbNewLine &
00759             & "Click ""OK"" to exit this application."
00760
00761             MsgBox sMsg, vbInformation, nvMSGBOXCAPTION
00762
00763         Else
00764             Call InitiateSystemShutdown(vbNullString, vbNullString, 0, API_FALSE, API_TRUE)
00765         End If
00766     End If
00767
00768     mRetVal = eCancel
00769
00770     Unload Me
00771
00772     Exit Sub
00773
00774 } Case ERROR_CHARS_NOT_SUPPORTED
00775     sMsg = "The MSDE setup failed because some of the characters entered for the Server Name are
00776     not supported by the current code page."
00777     mRetVal = eError
00778
00779 } Case ERROR_DUP_INSTANCE_NAME
00780     sMsg = "The MSDE setup failed because there is already a server named " & m_sServerName & " on
00781     this system."
00782     mRetVal = eError
00783
00784 } Case ERROR_INVALID_INSTANCE_SYNTAX
00785     sMsg = "The MSDE setup failed because some of the characters used in the Server Name are not
00786     valid for use in a SQL Server name."
00787     mRetVal = eError
00788
00789 } Case ERROR_OVER_MAX_INSTANCES
00790     sMsg = "The MSDE setup failed because there are already 16 instances of the SQL Server desktop
00791     edition installed on this system."
00792     mRetVal = eError
00793
00794 } Case ERROR_INSTALL_FAILURE
00795     sMsg = "The MSDE setup failed."
00796     mRetVal = eError
00797
00798 } Case Else
00799     sMsg = "The MSDE setup failed."
00800     mRetVal = eError
00801
00802 End Select
00803
00804 If Len(sMsg) Then MsgBox sMsg, vbInformation, nvMSGBOXCAPTION

```

```

00797 1 2 3 4 5
00798
00799     } Else
00800     m_RetVal = eError
00801     End If
00802
00803     } Else
00804     m_RetVal = eError
00805     End If
00806     } Else
00807     m_RetVal = eError
00808
00809     MsgBox "Unable to obtain available SQL MSI instance file.", vbInformation, nvMSGBOXCAPTION
00810     End If
00811     } Else
00812     m_RetVal = eError
00813     End If
00814
00815
00816     If hMSIErrLib Then
00817         Call FreeLibrary(hMSIErrLib)
00818         If bDeleteMSIErrFile Then Call DeleteFile(sTempPath)
00819     End If
00820
00821     If m_RetVal = eError Then
00822         lblInstal lMessage.Caption = "Errors occurred during the install process." & vbNewLine & vbNewLine &
>         "Unable to continue with server configuration."
00823         cmdNav(eCancel).Enabled = True
00824         cmdNav(eBack).Enabled = True
00825     End If
00826
00827 End Sub

```

```

00828
00829 Private Function DisableCancelButton(ByVal nProcessID&) As Boolean
00830
00831     Dim hDialog&, hButton&, nStyle&
00832
00833     hDialog = FindWindow("#32770", 0&)
00834
00835     If hDialog Then
00836         If nProcessID = ProcessIDFromhWnd(hDialog) Then
00837             hButton = FindWindowEx(hDialog, 0&, "Button", "Cancel")
00838
00839             If hButton Then
00840                 nStyle = GetWindowLong(hButton, GWL_STYLE)
00841
00842                 If (nStyle And WS_VISIBLE) <> 0 Then
00843                     Call SetWindowLong(hButton, GWL_STYLE, nStyle Xor WS_VISIBLE)
00844                     Call EnableWindow(hButton, API_FALSE)
00845
00846                     Call MoveWindow(hButton, -50&, -50&, 0&, 0&, API_TRUE)
00847
00848                     Call SetWindowPos(hDialog, HWND_TOPMOST, 0&, 0&, 0&, 0&, _
00849                         SWP_FRAMECHANGED Or SWP_NOMOVE Or _
00850                         SWP_NOREPOSITION Or SWP_NOSIZE Or SWP_SHOWWINDOW)
00851                     Call InvalidateRect(hDialog, 0&, API_TRUE)
00852                     Call SendMessage(hDialog, WM_PAINT, ByVal 0&, ByVal 0&)
00853                 End If
00854
00855                 DisableCancelButton = True
00856             End If
00857         End If
00858     End If
00859
00860 End Function

```

```

00861
00862 Private Function ProcessIDFromhWnd(ByVal hWnd&)
00863
00864     Dim nProcessID
00865
00866     Call GetWindowThreadProcessId(hWnd, nProcessID)
00867
00868     ProcessIDFromhWnd = nProcessID
00869
00870 End Function
00871

```

```

00872 Option Explicit
00873 ' demo project showing how to create a custom MSDE install
00874 ' by Bryan Stafford of New Vision Software - newvision@mmps.org
00875 ' this demo is released into the public domain "as is" without
00876 ' warranty or guaranty of any kind. In other words, use at
00877 ' your own risk.
00878 '
00879 ' Copyright © 2002 Bryan Stafford/New Vision Software
00880 '
00881 ' the code in this demo is copyrighted and may only be used in
00882 ' accordance with the rules set forth in the "House Rules"
00883 ' section of the web page found at http://www.mmps.org/vbvision/
00884 '
00885 ' see the general declararions section of MMain.bas for more info
00886 ' on this project
00887
00888 Private m_rc As RECT
00889 Private mRetVal As NAVIGATE_FLAGS

```

```

00891
00892 Public Function DisplayDialog(oPicture As StdPicture, ByVal IpRect&) As NAVIGATE_FLAGS
00893
00894 ' set the wait cursor in case loading the form takes a while
00895 Screen.MousePointer = vbHourglass
00896
00897 Dim hSysMenu&
00898
00899 ' first thing to do is get the handle to the system menu for this form
00900 hSysMenu = GetSystemMenu(hWnd, False)
00901
00902 Call DeleteMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
00903
00904 Set Picture = oPicture
00905
00906 CopyMemory m_rc, ByVal IpRect, Len(m_rc)
00907
00908 With m_rc
00909     Call SetWindowPos(hWnd, 0&, .Left, .Top, 0&, 0&, SWP_NOSIZE Or SWP_NOZORDER)
00910 End With
00911
00912 Screen.MousePointer = vbDefault
00913
00914 Show vbModal, MMain.TaskbarProxy
00915
00916
00917 CopyMemory ByVal IpRect, m_rc, Len(m_rc)
00918
00919 DisplayDialog = mRetVal
00920
00921
00922 End Function

```

```

00923
00924 Private Sub cmdNav_Click(Index As Integer)
00925
00926     Select Case Index
00927     Case eHelp
00928
00929     Case eCancel
00930         If AskAboutCancel() Then
00931             mRetVal = Index
00932         Unload Me
00933         End If
00934     Case eFinish
00935         Unload Me
00936     Case Else
00937         mRetVal = Index
00938     Unload Me
00939     End Select
00940
00941
00942 End Sub

```

```

00946
00947 Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
00948     If mRetVal = 0 Then Cancel = True
00949 End Sub

```

```

00950
00951 Private Sub Form_Unload(Cancel As Integer)
00952
00953     Call GetWindowRect(hWnd, m_rc)
00954
00955 End Sub
-----
00956
00957 Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
00958     ' keep the user from using Alt+F4 to close the form. notice
00959     ' that we have the KeyPreview property for the form set to True
00960     If KeyCode = vbKeyF4 Then KeyCode = 0
00961 End Sub
-----
00962
00963 Private Sub cmdInstallMSI_Click()
00964     tmrInstallMSI.Enabled = True
00965 End Sub
-----
00966
00967 Private Sub tmrProgress_Timer()
00968
00969     With pbProgress
00970         Select Case .CurrentPercentage
00971             Case Is < .MaxPercentage: .IncrementProgress
00972             Case Is >= .MaxPercentage: .IncrementProgress True
00973         End Select
00974     End With
00975
00976     'Refresh
00977
00978 End Sub
-----
00979
00980 Private Sub tmrInstallMSI_Timer()
00981
00982     tmrInstallMSI.Enabled = False
00983
00984     cmdNav(eCancel).Enabled = False
00985
00986     frmInstallMessages(1).Visible = True
00987     frmInstallMessages(0).Visible = False
00988
00989
00990     Select Case InstallMSI()
00991     Case eRebootNeeded
00992         If MsgBox("The Microsoft Installer application was installed successfully." & vbNewLine & "The system
00993             must be rebooted before the changes will take effect." _
00994             & vbNewLine & vbNewLine & "Would you like to reboot your system now?", vbYesNo Or vbQuestion,
00995             nvMSGBOXCAPTION) = vbYes Then
00996
00997             Call SetRunKey("/MSI", vbNullString)
00998
00999             mRetVal = eCancel
01000
01001             Call ExitWindowsEx(EWX_REBOOT Or EWX_FORCEIFHUNG, 0&)
01002
01003             Unload Me
01004         Else
01005             MsgBox "The install cannot be completed until the system is rebooted." & vbNewLine & vbNewLine _
01006             & "Please run the PAS Database Install application again after rebooting your system.",
01007             vbInformation, nvMSGBOXCAPTION
01008
01009             lblInstallInfo(2).Caption = "Cannot continue without reboot." & vbNewLine & vbNewLine & "Click "
01010             & "Finish" to exit."
01011
01012             mRetVal = eCancel
01013
01014             cmdNav(eFinish).Enabled = True
01015         End If
01016     Case eMSIInstallFailed
01017         MsgBox "The MSI installation failed." & vbNewLine & vbNewLine _
01018         & "Please run the PAS Database Install application again after rebooting your system.",
01019         vbInformation, nvMSGBOXCAPTION
01020
01021         lblInstallInfo(2).Caption = "Unable to continue." & vbNewLine & vbNewLine & "Click ""Finish"" to exit."

```

```

1 2
01021     m_RetVal = eCancel
01022
01023     cmdNav(eFinish).Enabled = True
01024
01025     } Case Else
01026     lblInstallInfo(2).Caption = "The Microsoft Installer application was successfully installed." &
    > vbNewLine & vbNewLine & "Click ""Next"" to continue."
01027
01028     cmdNav(eNext).Enabled = True
01029
01030     End Select
01031
01032 End Sub

```

```

01033
01034 Public Function InstallMSI() As MSI_REBOOT
01035
01036     Dim sMSIDir$, sMSIPath$, sCmdLine$, nWaitState&, nExitCode&, udtStartInf As STARTUPINFO, udtProcInf As
    > PROCESS_INFORMATION
01037
01038     On Error GoTo EH
01039
01040     Screen.MousePointer = vbHourglass
01041
01042     sMSIPath = AddBackslashToPath(App.Path)
01043
01044     sMSIDir = sMSIPath & "MSI"
01045     sMSIPath = sMSIPath & "MSI\"
01046
01047     Select Case GetOSVersion()
01048     Case Is < [eNT4.0]
01049         'sMSIDir = sMSIPath & "A"
01050         sMSIPath = sMSIPath & "INSTMSI.EXE" ' "A\msiinst.exe"
01051
01052     Case Else
01053         'sMSIDir = sMSIPath & "W"
01054         sMSIPath = sMSIPath & "INSTMSIW.EXE" ' "W\msiinst.exe"
01055
01056     End Select
01057
01058     ' c:\test\msiinst.exe /i instmsi.msi /qb+
01059
01060     sCmdLine = " /c:""msiinst.exe /i instmsi.msi /qn"" ' "/c:""instmsi.msi /delayreboot"" ' "/0" '
    > "/c:""instmsi.msi /delayreboot"" ' /c:"instmsi.msi /delayreboot"
01061
01062     'sMSIPath = Chr$(34) & sMSIPath & Chr$(34)
01063
01064     sMSIPath = ReturnShortPathName(sMSIPath)
01065     sMSIDir = ReturnShortPathName(sMSIDir)
01066
01067     With udtStartInf
01068         .cb = Len(udtStartInf)
01069         .lpTitle = vbNullChar
01070     End With
01071
01072     'If CreateProcess(vbNullString, sExePath, ByVal 0&, ByVal 0&, 0&, CREATE_UNICODE_ENVIRONMENT, ByVal
    > StrPtr(sGetEnvVar), sMSIDir, udtStartInf, udtProcInf) Then
01073
01074     If CreateProcess(sMSIPath, sCmdLine, 0&, 0&, API_FALSE, NORMAL_PRIORITY_CLASS, ByVal 0&, sMSIDir,
    > udtStartInf, udtProcInf) Then
01075         Do
01076
01077             With pbProgress
01078                 Select Case .CurrentPercentage
01079                 Case Is < .MaxPercentage: .IncrementProgress
01080                 Case Is >= .MaxPercentage: .IncrementProgress True
01081                 End Select
01082             End With
01083
01084             Yield
01085             nWaitState = WaitForSingleObject(udtProcInf.hProcess, 500&)
01086         Loop Until (nWaitState = WAIT_OBJECT_0) Or (nWaitState = WAIT_FAILED)
01087
01088         Call GetExitCodeProcess(udtProcInf.hProcess, nExitCode)
01089
01090         Call CloseHandle(udtProcInf.hThread)
01091         Call CloseHandle(udtProcInf.hProcess)
01092
01093     With pbProgress
01094     1 2 3     .ProgressUnits = 1

```

```
01095 1 2 3 .IncrementProgress
01096     End With
01097
01098
01099     Select Case nExitCode
01100     { Case ERROR_SUCCESS: InstallMSI = eRebootNotNeeded
01101     { Case ERROR_SUCCESS_REBOOT_REQUIRED: InstallMSI = eRebootNeeded
01102     { Case Else: InstallMSI = eMSIInstallFailed
01103     End Select
01104
01105     Else
01106     InstallMSI = eMSIInstallFailed
01107     End If
01108
01109
01110     Screen.MousePointer = vbDefault
01111
01112 ExitNow:
01113     Exit Function
01114
01115 EH:
01116     Screen.MousePointer = vbDefault
01117
01118     With Err
01119     » MsgBox "The following error occurred during the MSI install: " & .Description & " " & CStr(.Number),
01120         vbExclamation, nvMSGBOXCAPTION
01121     .Clear
01122     End With
01123
01124     InstallMSI = eMSIInstallFailed
01125
01126     Resume ExitNow
01127
01128 End Function
01129
01130
01131
```

```

01132 Option Explicit
01133 ' demo project showing how to create a custom MSDE install
01134 ' by Bryan Stafford of New Vision Software - newvision@myps.org
01135 ' this demo is released into the public domain "as is" without
01136 ' warranty or guaranty of any kind. In other words, use at
01137 ' your own risk.
01138 '
01139 ' Copyright © 2002 Bryan Stafford/New Vision Software
01140 '
01141 ' the code in this demo is copyrighted and may only be used in
01142 ' accordance with the rules set forth in the "House Rules"
01143 ' section of the web page found at http://www.myps.org/vbvision/
01144 '
01145 ' see the general declararions section of MMain.bas for more info
01146 ' on this project
01147
01148
01149 Private m_rc As RECT
01150 Private mRetVal As NAVIGATE_FLAGS

```

```

01151
01152 Public Function DisplayDialog(oPicture As StdPicture, ByVal IpRect&) As NAVIGATE_FLAGS
01153
01154 ' set the wait cursor in case loading the form takes a while
01155 Screen.MousePointer = vbHourglass
01156
01157 Dim hSysMenu&
01158
01159 ' first thing to do is get the handle to the system menu for this form
01160 hSysMenu = GetSystemMenu(hWnd, False)
01161
01162 Call DeleteMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
01163
01164
01165 Set Picture = oPicture
01166
01167 CopyMemory m_rc, ByVal IpRect, Len(m_rc)
01168
01169 With m_rc
01170     Call SetWindowPos(hWnd, 0&, .Left, .Top, 0&, 0&, SWP_NOSIZE Or SWP_NOZORDER)
01171 End With
01172
01173
01174 lblWarning.Caption = "Portions of the following Microsoft install application may take A VERY LONG TIME"
> & vbNewLine _
01175 & "to complete and, at times, it may appear as if the application has stalled." & vbNewLine &
> vbNewLine _
01176 & "Please be patient and do not attempt to terminate the process."
01177
01178
01179 Screen.MousePointer = vbDefault
01180
01181 Show vbModal, MMain.TaskbarProxy
01182
01183
01184 CopyMemory ByVal IpRect, m_rc, Len(m_rc)
01185
01186
01187 DisplayDialog = mRetVal
01188
01189 End Function

```

```

01190
01191 Private Sub cmdNav_Click(Index As Integer)
01192
01193     Select Case Index
01194     Case eHelp
01195
01196     Case eCancel
01197         If AskAboutCancel() Then
01198             mRetVal = Index
01199
01200             Unload Me
01201         End If
01202
01203     Case Else
01204         mRetVal = Index
01205
01206         Unload Me
01207     End Select

```

```
01208 1
01209 End Sub
-----
01210
01211 Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
01212     If mRetVal = 0 Then Cancel = True
01213 End Sub
-----
01214
01215 Private Sub Form_Unload(Cancel As Integer)
01216     Call GetWindowRect(hWnd, m_rc)
01217 End Sub
-----
01218
01219 Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
01220     ' keep the user from using Alt+F4 to close the form. notice
01221     ' that we have the KeyPreview property for the form set to True
01222     If KeyCode = vbKeyF4 Then KeyCode = 0
01223 End Sub
-----
```

```
01224 Option Explicit
01225 ' demo project showing how to create a custom MSDE install
01226 ' by Bryan Stafford of New Vision Software - newvision@myps.org
01227 ' this demo is released into the public domain "as is" without
01228 ' warranty or guaranty of any kind. In other words, use at
01229 ' your own risk.
01230 '
01231 ' Copyright © 2002 Bryan Stafford/New Vision Software
01232 '
01233 ' the code in this demo is copyrighted and may only be used in
01234 ' accordance with the rules set forth in the "House Rules"
01235 ' section of the web page found at http://www.myps.org/vbvision/
01236 '
01237 ' see the general declararions section of MMain.bas for more info
01238 ' on this project
01239
01240
```

```

01241 Option Explicit
01242 ' demo project showing how to create a custom MSDE install
01243 ' by Bryan Stafford of New Vision Software - newvision@mvps.org
01244 ' this demo is released into the public domain "as is" without
01245 ' warranty or guaranty of any kind. In other words, use at
01246 ' your own risk.
01247 '
01248 ' Copyright © 2002 Bryan Stafford/New Vision Software
01249 '
01250 ' the code in this demo is copyrighted and may only be used in
01251 ' accordance with the rules set forth in the "House Rules"
01252 ' section of the web page found at http://www.mvps.org/vbvision/
01253 '
01254 ' see the general declararions section of MMain.bas for more info
01255 ' on this project
01256
01257 Private m_rc As RECT
01258 Private mRetVal As NAVIGATE_FLAGS

```

```

01260
01261 Public Function DisplayDialog(oPicture As StdPicture, ByVal IpRect&) As NAVIGATE_FLAGS
01262
01263 ' set the wait cursor in case loading the form takes a while
01264 Screen.MousePointer = vbHourglass
01265
01266 Dim hSysMenu&
01267
01268 ' first thing to do is get the handle to the system menu for this form
01269 hSysMenu = GetSystemMenu(hWnd, False)
01270
01271 Call DeleteMenu(hSysMenu, SC_CLOSE, MF_BYCOMMAND)
01272
01273
01274 If oPicture Is Nothing Then Set oPicture = CreateTitlePicture(hWnd)
01275
01276 Set Picture = oPicture
01277
01278
01279 CopyMemory m_rc, ByVal IpRect, Len(m_rc)
01280
01281 With m_rc
01282 Call SetWindowPos(hWnd, 0&, .Left, .Top, 0&, 0&, SWP_NOSIZE Or SWP_NOZORDER)
01283 End With
01284
01285
01286 Label1(1).Caption = "This application allows you to install and configure a new MSDE SQL Server
> instance." & vbNewLine & vbNewLine _
01287 & "It is HIGHLY recommended that you close all other applications, including any anti-virus
> applications that may" _
01288 & " be running in the background, before you continue." & vbNewLine & vbNewLine & vbNewLine _
01289 & "Click ""Next"" to begin the installation process."
01290
01291
01292 Screen.MousePointer = vbDefault
01293
01294 Show vbModal, MMain.TaskbarProxy
01295
01296 CopyMemory ByVal IpRect, m_rc, Len(m_rc)
01297
01298 DisplayDialog = mRetVal
01299
01300 End Function

```

```

01301
01302 Private Sub cmdNav_Click(Index As Integer)
01303
01304 Select Case Index
01305 Case eHelp
01306
01307 Case eCancel
01308 If AskAboutCancel() Then
01309 mRetVal = Index
01310
01311 Unload Me
01312 End If
01313
01314 Case Else
01315 mRetVal = Index
01316

```

```
01317 1 2 Unload Me
01318   End Select
01319 End Sub
01320


---


01321
01322 Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
01323     If m_RetVal = 0 Then Cancel = True
01324 End Sub


---


01325
01326 Private Sub Form_Unload(Cancel As Integer)
01327     Call GetWindowRect(hWnd, m_rc)
01328
01329 End Sub


---


01331
01332 Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
01333     ' keep the user from using Alt+F4 to close the form. notice
01334     ' that we have the KeyPreview property for the form set to True
01335     If KeyCode = vbKeyF4 Then KeyCode = 0
01336 End Sub


---


```

```

01337 Option Explicit
01338 ' demo project showing how to create a custom MSDE install
01339 ' by Bryan Stafford of New Vision Software - newvision@myps.org
01340 ' this demo is released into the public domain "as is" without
01341 ' warranty or guaranty of any kind. In other words, use at
01342 ' your own risk.
01343 '
01344 ' Copyright © 2002 Bryan Stafford/New Vision Software
01345 '
01346 ' the code in this demo is copyrighted and may only be used in
01347 ' accordance with the rules set forth in the "House Rules"
01348 ' section of the web page found at http://www.myps.org/vbvision/
01349 '
01350 ' see the general declararions section of MMain.bas for more info
01351 ' on this project
01352
01353
01354 Public Const API_FALSE As Long = 0&
01355 Public Const API_TRUE As Long = 1&
01356 Public Const MAX_PATH As Long = 260&
01357
01358 Public Const ERROR_SUCCESS As Long = 0&
01359 Public Const ERROR_INSTALL_FAILURE As Long = 1603&
01360 Public Const ERROR_SUCCESS_REBOOT_INITIATED As Long = 1641&
01361 Public Const ERROR_SUCCESS_REBOOT_REQUIRED As Long = 3010&
01362 Public Const NOERROR As Long = 0&
01363
01364 Public Const BST_CHECKED As Long = &H1&
01365
01366 Public Const EWX_FORCEIFHUNG As Long = 10&
01367
01368 Public Const nvMSGBOXCAPTION As String = "Custom Database Configuration"
01369
01370 Public Type RECT
01371     Left As Long
01372     Top As Long
01373     Right As Long
01374     Bottom As Long
01375 End Type
01376
01377 Public Type POINTAPI
01378     x As Long
01379     y As Long
01380 End Type
01381
01382 Public Type MSG
01383     hWnd As Long
01384     message As Long
01385     wParam As Long
01386     lParam As Long
01387     time As Long
01388     pt As POINTAPI
01389 End Type
01390
01391 Public Type FILETIME
01392     dwLowDateTime As Long
01393     dwHighDateTime As Long
01394 End Type
01395
01396 Public Type WIN32_FIND_DATA_A
01397     dwFileAttributes As Long
01398     ftCreationTime As Currency
01399     ftLastAccessTime As Currency
01400     ftLastWriteTime As Currency
01401     nFileSizeHigh As Long
01402     nFileSizeLow As Long
01403     dwReserved0 As Long
01404     dwReserved1 As Long
01405     cFileName(0 To 259) As Byte
01406     cAlternateFileName(0 To 13) As Byte
01407 End Type
01408
01409 Public Enum SQL_SERVER_ROLLS
01410     eSystem_Administrators = 1&
01411     eSecurity_Administrators = 2&
01412     eServer_Administrators = 3&
01413     eSetup_Administrators = 4&
01414     eProcess_Administrators = 5&
01415     eDisk_Administrators = 6&

```

```

01416 | eDatabase_Creators = 7&
01417 | eBulk_Insert_Administrators = 8&
01418 | End Enum
01419
01420 | Public Type GUID
01421 |     Data1 As Long
01422 |     Data2 As Integer
01423 |     Data3 As Integer
01424 |     Data4(7) As Byte
01425 | End Type
01426
01427 | Public Type PICTDESC_BMP
01428 |     Size As Long
01429 |     Type As Long
01430 |     hBmp As Long
01431 |     hPal As Long
01432 |     Reserved As Long
01433 | End Type
01434
01435 | Public Declare Function OleCreatePictureIndirect Lib "ole32.dll" (PicDesc As PICTDESC_BMP, RefIID As
> | GUID, ByVal fPictureOwnsHandle As Long, IPic As IPicture) As Long
01436
01437 | Public Declare Function SaveDC Lib "gdi32" (ByVal hDC&) As Long
01438
01439 | Public Declare Function RestoreDC& Lib "gdi32" (ByVal hDC&, ByVal nSavedDC&)
01440
01441 | Public Declare Function LoadBitmapBynum& Lib "user32" Alias "LoadBitmapA" (ByVal hInstance&, ByVal
> | lpBitmapName&)
01442
01443
01444 | Public Type BITMAP '14 bytes
01445 |     bmType As Long
01446 |     bmWidth As Long
01447 |     bmHeight As Long
01448 |     bmWidthBytes As Long
01449 |     bmPlanes As Integer
01450 |     bmBitsPixel As Integer
01451 |     bmBits As Long
01452 | End Type
01453
01454 | Public Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" (ByVal hObject As Long, ByVal nCount As
> | Long, lpObject As Any) As Long
01455
01456 | Public Declare Function TranslateMessage Lib "user32" (lpMsg As MSG) As Long
01457 | Public Declare Function DispatchMessage Lib "user32" Alias "DispatchMessageA" (lpMsg As MSG) As Long
01458 | Public Declare Function PeekMessage Lib "user32" Alias "PeekMessageA" (lpMsg As MSG, ByVal hWnd&, ByVal
> | wParamFilterMin&, ByVal wParamFilterMax&, ByVal wRemoveMsg&) As Long
01459
01460 | Public Declare Function GetProcAddress Lib "kernel32" (ByVal hModule&, ByVal lpProcName$) As Long
01461
01462 | Public Declare Function FillRect Lib "user32" (ByVal hDC&, lpRect As RECT, ByVal hBrush&) As Long
01463
01464 | ' LoadImage constants
01465 | Public Const IMAGE_BITMAP As Long = 0& ' <- loads a bitmap
01466 | Public Const IMAGE_ICON As Long = 1& ' <- loads an icon
01467 | Public Const IMAGE_CURSOR As Long = 2& ' <- loads a cursor
01468 | Public Const LR_DEFAULTCOLOR As Long = &H0& ' <- default value
01469 | Public Const LR_SHARED As Long = &H8000& ' <- use this in the fuLoad param to share the image handle
01470 | Public Const LR_LOADFROMFILE = &H10 ' <- use this in the fuLoad param to load a graphic
01471 | ' from a file. you will have to change lpszName to a string param to pass the file name
01472 | ' to the function.
01473
01474 | ' alias used to load from resource ID number instead of string
01475 | Public Declare Function LoadImageBynum Lib "user32" Alias "LoadImageA" (ByVal hInst&, ByVal lpszName&, _
01476 |     ByVal uType&, ByVal cxDesired&, ByVal cyDesired&, ByVal fuLoad&) As Long
01477
01478 | ' used to destroy icon handles when we've finished with them
01479 | Public Declare Function DestroyIcon Lib "user32" (ByVal hIcon&) As Long
01480
01481 | Public Declare Function SetFocusAPI Lib "user32" Alias "SetFocus" (ByVal hWnd&) As Long
01482
01483
01484 | Public Const ERROR_CHARS_NOT_SUPPORTED As Long = 50035
01485 | Public Const ERROR_DUP_INSTANCE_NAME As Long = 50043
01486 | Public Const ERROR_INVALID_INSTANCE_SYNTAX As Long = 50047
01487 | Public Const ERROR_CONFIGURE_SERVER_FAILURE As Long = 60001
01488 | Public Const ERROR_INSTALL_CATALOG_STP_FAILURE As Long = 60002
01489 | Public Const ERROR_INSTALL_DTC_FAILURE As Long = 60003
01490 | Public Const ERROR_INSTALL_PERFMON_FAILURE As Long = 60004

```

```

01491 Public Const ERROR_INSTALL_SQLAGENT_SECURITY_FAILURE As Long = 60005
01492 Public Const ERROR_INSTALL_SQLREDIS_FAILURE As Long = 60006
01493 Public Const ERROR_OVER_MAX_INSTANCES As Long = 60007
01494 Public Const ERROR_UPGRADE_FAILURE As Long = 60009
01495
01496
01497 Public Declare Function RetrieveMSIExitCode Lib "nvMSI_ERR" () As Long
01498 Public Declare Function InitializeMSILib Lib "nvMSI_ERR" () As Long
01499
01500
01501 Public Type DLLVERSIONINFO
01502     cbSize As Long
01503     dwMajorVersion As Long
01504     dwMinorVersion As Long
01505     dwBuildNumber As Long
01506     dwPlatformId As Long
01507 End Type
01508
01509 Public Declare Function DllGetVersion Lib "msi" (pDVI As DLLVERSIONINFO) As Long
01510
01511
01512 Public Type STARTUPINFO
01513     cb As Long
01514     lpReserved As Long
01515     lpDesktop As Long
01516     lpTitle As String
01517     dwX As Long
01518     dwY As Long
01519     dwXSize As Long
01520     dwYSize As Long
01521     dwXCountChars As Long
01522     dwYCountChars As Long
01523     dwFillAttribute As Long
01524     dwFlags As Long
01525     wShowWindow As Integer
01526     cbReserved2 As Integer
01527     lpReserved2 As Long
01528     hStdInput As Long
01529     hStdOutput As Long
01530     hStdError As Long
01531 End Type
01532
01533 Public Type PROCESS_INFORMATION
01534     hProcess As Long
01535     hThread As Long
01536     dwProcessId As Long
01537     dwThreadId As Long
01538 End Type
01539
01540 ' dwCreationFlag values CREATE_NEW_PROCESS_GROUP
01541 Public Const CREATE_DEFAULT_ERROR_MODE As Long = &H4000000
01542 Public Const CREATE_NEW_CONSOLE = &H10
01543 Public Const CREATE_NEW_PROCESS_GROUP = &H200
01544 Public Const CREATE_SEPARATE_WOW_VDM As Long = &H800&
01545 Public Const CREATE_SHARED_WOW_VDM As Long = &H1000&
01546 Public Const CREATE_SUSPENDED = &H4
01547 Public Const CREATE_UNICODE_ENVIRONMENT As Long = &H400&
01548 Public Const DEBUG_PROCESS = &H1
01549 Public Const DEBUG_ONLY_THIS_PROCESS = &H2
01550 Public Const DETACHED_PROCESS = &H8
01551
01552 Public Declare Function CreateProcess Lib "kernel32" Alias "CreateProcessA" (ByVal lpApplicationName$,
    »   ByVal lpCommandLine$, ByVal lpProcessAttributes&, ByVal lpThreadAttributes&, ByVal bInheritHandles&, ByVal
    »   dwCreationFlags&, lpEnvironment As Any, ByVal lpCurrentDirectory$, lpStartupInfo As STARTUPINFO,
    »   lpProcessInformation As PROCESS_INFORMATION) As Long
01553 Public Declare Function CloseHandle Lib "kernel32" (ByVal hObject&) As Long
01554
01555 Public Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName$, ByVal lpWindowName
    »   &) As Long
01556 Public Declare Function FindWindowStr Lib "user32" Alias "FindWindowA" (ByVal lpClassName$, ByVal
    »   lpWindowName$) As Long
01557 Public Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWndParent&, ByVal
    »   hWndChildAfter&, ByVal lpClassName$, ByVal lpWindowName$) As Long
01558
01559 Public Declare Function WaitForInputIdle Lib "user32" (ByVal hProcess&, ByVal dwMilliseconds&) As Long
01560
01561
01562 Public Const RDW_INVALIDATE As Long = &H1&
01563 Public Const RDW_INTERNALPAINT As Long = &H2&

```

```

01564 Public Const RDW_ERASE As Long = &H4&
01565
01566 Public Const RDW_VALIDATE As Long = &H8&
01567 Public Const RDW_NOINTERNALPAINT As Long = &H10&
01568 Public Const RDW_NOERASE As Long = &H20&
01569
01570 Public Const RDW_NOCHILDREN As Long = &H40&
01571 Public Const RDW_ALLCHILDREN As Long = &H80&
01572
01573 Public Const RDW_UPDATENOW As Long = &H100&
01574 Public Const RDW_ERASENOW As Long = &H200&
01575
01576 Public Const RDW_FRAME As Long = &H400&
01577 Public Const RDW_NOFRAME As Long = &H800&
01578
01579 Public Declare Function RedrawWindow Lib "user32" (ByVal hWnd&, ByVal lprcUpdate&, ByVal hrgnUpdate&, ByVal
» fuRedraw&) As Long
01580
01581 Public Declare Function GetFileSize Lib "kernel32" (ByVal hFile&, ByRef lpFileSizeHigh&) As Long
01582
01583 Public Const nvSCmp_ERROR_CONDITION As Long = 0& ' Error(s) ocured
01584
01585 Public Declare Function CompressString Lib "nvSCmp32.dll" (ByVal lpOutBuff&, ByVal nOutBuffLen&, ByVal
» lpInBuff&, ByVal nInBuffLen&) As Long
01586 Public Declare Function InflateString Lib "nvSCmp32.dll" (ByVal lpOutBuff&, ByVal nOutBuffLen&, ByVal
» lpInBuff&, ByVal nInBuffLen&) As Long
01587
01588 Public Declare Function InitCommonControls Lib "comctl32.dll" () As Long
01589
01590 Public Const S_OK As Long = &H0&
01591
01592 Public Declare Function SQLNS_DIRegisterServer Lib "sqlns" Alias "DIRegisterServer" () As Long
01593 Public Declare Function SQLLEX_DIRegisterServer Lib "sqllex" Alias "DIRegisterServer" () As Long
01594
01595 Public Declare Function InvalidateRect Lib "user32" (ByVal hWnd&, ByVal lpRect&, ByVal bErase&) As Long
01596
01597
01598 Public Declare Function strlenA Lib "kernel32" (ByVal lpString As Long) As Long
01599 Public Declare Function strlenW Lib "kernel32" (ByVal lpString As Long) As Long
01600
01601
01602 Public Declare Function GetLogicalDriveStrings Lib "kernel32" Alias "GetLogicalDriveStringsA" (ByVal
» nBufferLength As Long, ByVal lpBuffer As String) As Long
01603
01604 Public Const MOVEFILE_REPLACE_EXISTING As Long = &H1&
01605 Public Const MOVEFILE_COPY_ALLOWED As Long = &H2&
01606 Public Const MOVEFILE_DELAY_UNTIL_REBOOT As Long = &H4&
01607 Public Const MOVEFILE_WRITE_THROUGH As Long = &H8&
01608 Public Const MOVEFILE_CREATE_HARDLINK As Long = &H10&
01609 Public Const MOVEFILE_FAIL_IF_NOT_TRACKABLE As Long = &H20&
01610
01611 Public Const SC_CLOSE As Long = &HF060&
01612 Public Const MF_BYCOMMAND As Long = &H0&
01613
01614 Public Declare Function GetSystemMenu Lib "user32" (ByVal hWnd&, ByVal bRevert&) As Long
01615 Public Declare Function DeleteMenu Lib "user32" (ByVal hMenu&, ByVal nPosition&, ByVal wFlags&) As Long
01616
01617 Public Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (lpDest As Any, lpSource As Any, ByVal
» cBytes As Long)
01618
01619 ' SetWindowPos Flags
01620 Public Const SWP_NOSIZE As Long = &H1&
01621 Public Const SWP_NOMOVE As Long = &H2&
01622 Public Const SWP_NOZORDER As Long = &H4&
01623 Public Const SWP_NOREDRAW As Long = &H8&
01624 Public Const SWP_NOACTIVATE As Long = &H10&
01625 Public Const SWP_FRAMECHANGED As Long = &H20 ' The frame changed: send WM_NCCALCSIZE
01626 Public Const SWP_SHOWWINDOW As Long = &H40&
01627 Public Const SWP_HIDEWINDOW As Long = &H80&
01628 Public Const SWP_NOCOPYBITS As Long = &H100&
01629 Public Const SWP_NOOWNERZORDER As Long = &H200 ' Don't do owner Z ordering
01630
01631 Public Const SWP_DRAWFRAME As Long = SWP_FRAMECHANGED
01632 Public Const SWP_NOREPOSITION As Long = SWP_NOOWNERZORDER
01633
01634 ' SetWindowPos() hwndInsertAfter values
01635 Public Const HWND_TOP As Long = 0
01636 Public Const HWND_BOTTOM As Long = 1
01637 Public Const HWND_TOPMOST As Long = -1

```

```

01638 Public Const HWND_NOTOPMOST As Long = -2
01639
01640 Public Declare Function SetWindowPos Lib "user32" (ByVal hWnd&, ByVal hWndInsertAfter&, ByVal x&, ByVal y&,
»   ByVal cx&, ByVal cy&, ByVal wFlags&) As Long
01641
01642 Public Declare Function GetWindowRect Lib "user32" (ByVal hWnd&, lpRect As RECT) As Long
01643
01644 Public Const NORMAL_PRIORITY_CLASS As Long = &H20&
01645 Public Const WAIT_OBJECT_0 As Long = 0&
01646 Public Const WAIT_FAILED As Long = -1&
01647
01648 Public Const BM_GETCHECK As Long = &HF0&
01649 Public Const BM_SETCHECK As Long = &HF1&
01650
01651 Public Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd&, ByVal wParam, wParam As
»   Any, lParam As Any) As Long
01652 Public Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hWnd&, ByVal wParam, ByVal
»   lParam, lParam As Any) As Long
01653
01654 Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds&)
01655
01656 Public Const WM_CLOSE As Long = &H10&
01657 Public Const SW_HIDE As Long = 0&
01658
01659 Public Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle&, ByVal dwMilliseconds&) As Long
01660
01661 Public Declare Function GetDC Lib "user32" (ByVal hWnd&) As Long
01662 Public Declare Function CreateCompatibleDC Lib "gdi32" (ByVal hDC&) As Long
01663 Public Declare Function SelectObject Lib "gdi32" (ByVal hDC&, ByVal hObject&) As Long
01664 Public Declare Function CreateCompatibleBitmap Lib "gdi32" (ByVal hDC&, ByVal nWidth&, ByVal nHeight&) As
»   Long
01665 Public Declare Function ReleaseDC Lib "user32" (ByVal hWnd&, ByVal hDC&) As Long
01666 Public Declare Function DeleteDC Lib "gdi32" (ByVal hDC&) As Long
01667 Public Declare Function DeleteObject Lib "gdi32" (ByVal hObject&) As Long
01668
01669 Public Declare Function GetClientRect Lib "user32" (ByVal hWnd&, lpRect As RECT) As Long
01670
01671
01672 Public Declare Function LoadLibrary Lib "kernel32" Alias "LoadLibraryA" (ByVal lpFileName$) As Long
01673 Public Declare Function FreeLibrary Lib "kernel32" (ByVal hLibModule&) As Long
01674
01675 Public Type PALETTEENTRY
01676     peRed As Byte
01677     peGreen As Byte
01678     peBlue As Byte
01679     peFlags As Byte
01680 End Type
01681
01682 Public Type LOGPALETTE256
01683     palVersion As Integer
01684     palNumEntries As Integer
01685     palPalEntry(255) As PALETTEENTRY
01686 End Type
01687
01688 Public Const RASTERCAPS As Long = 38&
01689 Public Const SIZEPALETTE As Long = 104&
01690 Public Const RC_PALETTE As Long = &H100&
01691
01692 Public Declare Function GetDeviceCaps Lib "gdi32" (ByVal hDC&, ByVal nIndex&) As Long
01693 Public Declare Function GetSystemPaletteEntries Lib "gdi32" (ByVal hDC&, ByVal wStartIndex&, ByVal
»   wNumEntries&, lpPaletteEntries As PALETTEENTRY) As Long
01694 Public Declare Function CreatePalette Lib "gdi32" (lpLogPalette As LOGPALETTE256) As Long
01695 Public Declare Function SelectPalette Lib "gdi32" (ByVal hDC&, ByVal hPalette&, ByVal bForceBackground&) As
»   Long
01696 Public Declare Function BitBlt Lib "gdi32" (ByVal hDestDC&, ByVal x&, ByVal y&, ByVal nWidth&, ByVal
»   nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&) As Long
01697
01698 Public Declare Function RealizePalette Lib "gdi32" (ByVal hDC&) As Long
01699
01700 ' Message Function Templates
01701 Public Declare Function GetMessage Lib "user32" Alias "GetMessageA" (lpMsg As MSG, ByVal hWnd&, ByVal
»   wParamFilterMin&, ByVal wParamFilterMax&) As Long
01702
01703 ' PeekMessage() Options
01704 Public Const PM_NOREMOVE As Long = &H0
01705 Public Const PM_REMOVE As Long = &H1
01706 Public Const PM_NOYIELD As Long = &H2
01707
01708 Public Declare Function CreateBitmap Lib "gdi32" (ByVal nWidth&, ByVal nHeight&, ByVal nPlanes&, ByVal

```

```

» nBitsCount&, lpBits As Any) As Long
01709 Public Declare Function SetBkColor Lib "gdi32" (ByVal hdc&, ByVal crColor&) As Long
01710
01711 Public Type OSVERSIONINFO
01712     dwOSVersionInfoSize As Long
01713     dwMajorVersion As Long
01714     dwMinorVersion As Long
01715     dwBuildNumber As Long
01716     dwPlatformId As Long
01717     szCSDVersion As String * 128 ' Maintenance string for PSS usage
01718 End Type
01719
01720 Public Declare Function GetVersionEx Lib "kernel32" Alias "GetVersionExA" (lpVersionInformation As
» OSVERSIONINFO) As Long
01721
01722 Public Declare Function GetShortPathName Lib "kernel32" Alias "GetShortPathNameA" (ByVal lpszLongPath$,
» ByVal lpszShortPath$, ByVal cchBuffer&) As Long
01723
01724 Public Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA" (ByVal lpBuffer$, nSize&)
» As Long
01725
01726 Public Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" (ByVal lpFileName$,
» lpFindFileData As WIN32_FIND_DATA_A) As Long
01727 Public Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" (ByVal hFindFile&, lpFindFileData
» As WIN32_FIND_DATA_A) As Long
01728 Public Declare Function GetFileAttributes Lib "kernel32" Alias "GetFileAttributesA" (ByVal lpFileName$) As
» Long
01729 Public Declare Function FindClose Lib "kernel32" (ByVal hFindFile&) As Long
01730
01731 Public Const INVALID_HANDLE_VALUE As Long = (-1&)
01732 Public Const ERROR_NO_MORE_FILES As Long = 18&
01733
01734 Public Declare Function GetModuleHandle Lib "kernel32" Alias "GetModuleHandleA" (ByVal lpModuleName$) As
» Long
01735
01736 Public Const FILE_GENERIC_READ = &H120089 ' (STANDARD_RIGHTS_READ Or FILE_READ_DATA Or
» FILE_READ_ATTRIBUTES Or FILE_READ_EA Or SYNCHRONIZE)
01737
01738
01739 Public Const FILE_GENERIC_WRITE = &H120114 ' (STANDARD_RIGHTS_WRITE Or FILE_WRITE_DATA Or
» FILE_WRITE_ATTRIBUTES Or FILE_WRITE_EA Or FILE_APPEND_DATA Or SYNCHRONIZE)
01740
01741
01742 Public Const FILE_GENERIC_EXECUTE = &H1200A0 ' (STANDARD_RIGHTS_EXECUTE Or FILE_READ_ATTRIBUTES Or
» FILE_EXECUTE Or SYNCHRONIZE)
01743
01744 Public Const FILE_SHARE_READ = &H1
01745 Public Const FILE_SHARE_WRITE = &H2
01746 Public Const FILE_ATTRIBUTE_READONLY = &H1
01747 Public Const FILE_ATTRIBUTE_HIDDEN = &H2
01748 Public Const FILE_ATTRIBUTE_SYSTEM = &H4
01749 Public Const FILE_ATTRIBUTE_DIRECTORY = &H10
01750 Public Const FILE_ATTRIBUTE_ARCHIVE = &H20
01751 Public Const FILE_ATTRIBUTE_NORMAL = &H80
01752 Public Const FILE_ATTRIBUTE_TEMPORARY = &H100
01753 Public Const FILE_ATTRIBUTE_COMPRESSED = &H800
01754 Public Const FILE_NOTIFY_CHANGE_FILE_NAME = &H1
01755 Public Const FILE_NOTIFY_CHANGE_DIR_NAME = &H2
01756 Public Const FILE_NOTIFY_CHANGE_ATTRIBUTES = &H4
01757 Public Const FILE_NOTIFY_CHANGE_SIZE = &H8
01758 Public Const FILE_NOTIFY_CHANGE_LAST_WRITE = &H10
01759 Public Const FILE_NOTIFY_CHANGE_SECURITY = &H100
01760 Public Const MAILSLOT_NO_MESSAGE = (-1)
01761 Public Const MAILSLOT_WAIT_FOREVER = (-1)
01762 Public Const FILE_CASE_SENSITIVE_SEARCH = &H1
01763 Public Const FILE_CASE_PRESERVED_NAMES = &H2
01764 Public Const FILE_UNICODE_ON_DISK = &H4
01765 Public Const FILE_PERSISTENT_ACLS = &H8
01766 Public Const FILE_FILE_COMPRESSION = &H10
01767 Public Const FILE_VOLUME_IS_COMPRESSED = &H8000
01768 Public Const FS_CASE_IS_PRESERVED = &H2
01769 Public Const FS_CASE_SENSITIVE = &H1
01770 Public Const FS_UNICODE_STORED_ON_DISK = &H4
01771 Public Const FS_PERSISTENT_ACLS = &H8
01772 Public Const FS_FILE_COMPRESSION = &H10
01773 Public Const FS_VOL_IS_COMPRESSED = &H8000
01774
01775
01776 Public Const GENERIC_READ = &H80000000

```

```

01777 Public Const GENERIC_WRITE = &H40000000
01778 Public Const GENERIC_EXECUTE = &H20000000
01779 Public Const GENERIC_ALL = &H10000000
01780
01781 Public Const CREATE_NEW = 1
01782 Public Const CREATE_ALWAYS = 2
01783 Public Const OPEN_EXISTING = 3
01784 Public Const OPEN_ALWAYS = 4
01785 Public Const TRUNCATE_EXISTING = 5
01786
01787 Public Const FILE_FLAG_WRITE_THROUGH = &H80000000
01788 Public Const FILE_FLAG_OVERLAPPED = &H40000000
01789 Public Const FILE_FLAG_NO_BUFFERING = &H20000000
01790 Public Const FILE_FLAG_RANDOM_ACCESS = &H10000000
01791 Public Const FILE_FLAG_SEQUENTIAL_SCAN = &H80000000
01792 Public Const FILE_FLAG_DELETE_ON_CLOSE = &H40000000
01793 Public Const FILE_FLAG_BACKUP_SEMANTICS = &H20000000
01794 Public Const FILE_FLAG_POSIX_SEMANTICS = &H10000000
01795
01796 Public Type SYSTEMTIME
01797     wYear As Integer
01798     wMonth As Integer
01799     wDayOfWeek As Integer
01800     wDay As Integer
01801     wHour As Integer
01802     wMinute As Integer
01803     wSecond As Integer
01804     wMilliseconds As Integer
01805 End Type
01806
01807 Public Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName$, ByVal
    » dwDesiredAccess&, ByVal dwShareMode&, ByVal lpSecurityAttributes&, ByVal dwCreationDisposition&, ByVal
    » dwFlagsAndAttributes&, ByVal hTemplateFile&) As Long
01808
01809 Public Declare Function GetFileTime Lib "kernel32" (ByVal hFile&, lpCreationTime As FILETIME,
    » lpLastAccessTime As FILETIME, lpLastWriteTime As FILETIME) As Long
01810
01811 Public Declare Function FileTimeToLocalFileTime Lib "kernel32" (lpFileTime As FILETIME, lpLocalFileTime As
    » FILETIME) As Long
01812 Public Declare Function FileTimeToSystemTime Lib "kernel32" (lpFileTime As FILETIME, lpSystemTime As
    » SYSTEMTIME) As Long
01813
01814 Public Const EWX_LOGOFF As Long = 0
01815 Public Const EWX_SHUTDOWN As Long = 1
01816 Public Const EWX_REBOOT As Long = 2
01817 Public Const EWX_FORCE As Long = 4
01818
01819 Public Const READAPI As Long = 0 ' Flags for _lopen
01820 Public Const WRITEAPI As Long = 1
01821 Public Const READ_WRITE As Long = 2
01822
01823 Public Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags&, ByVal dwReserved&) As Long
01824
01825 Public Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess&, lpExitCode&) As Long
01826
01827 Public Declare Function GetSysColor Lib "user32" (ByVal nIndex&) As Long
01828 Public Declare Function CreateSolidBrush Lib "gdi32" (ByVal crColor&) As Long
01829
01830 Public Declare Function SetBkMode Lib "gdi32" (ByVal hdc&, ByVal nBkMode&) As Long
01831 Public Declare Function SetTextColor Lib "gdi32" (ByVal hdc&, ByVal crColor&) As Long
01832 Public Declare Function TextOut Lib "gdi32" Alias "TextOutA" (ByVal hdc&, ByVal x&, ByVal y&, ByVal
    » lpString$, ByVal nCount&) As Long
01833
01834 Public Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer$,
    » ByVal nSize&) As Long
01835
01836 Public Const GWL_STYLE As Long = (-16&)
01837 Public Const GWL_EXSTYLE As Long = (-20&)
01838
01839 Public Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hWnd&, ByVal nIndex&) As
    » Long
01840 Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&, ByVal nIndex&,
    » ByVal dwNewLong&) As Long
01841
01842 Public Declare Function EnableWindow Lib "user32" (ByVal hWnd&, ByVal fEnable&) As Long
01843
01844 Public Const WS_VISIBLE As Long = &H10000000
01845
01846 Public Declare Function MoveWindow Lib "user32" (ByVal hWnd&, ByVal x&, ByVal y&, ByVal nWidth&, ByVal

```

```

» nHeight&, ByVal bRepairnt&) As Long
01847
01848 Public Const WM_PAINT As Long = &HF&
01849
01850 Public Declare Function GetWindowThreadProcessId Lib "user32" (ByVal hWnd&, lpdwProcessId&) As Long
01851
01852 Public Const COLOR_BTNFACE As Long = 15&
01853
01854 Public Declare Function GetModuleFileName Lib "kernel32" Alias "GetModuleFileNameA" (ByVal hModule&, ByVal
» lpFileName$, ByVal nSize&) As Long
01855
01856 Public Declare Function GetTempPath Lib "kernel32" Alias "GetTempPathA" (ByVal nBufferLength&, ByVal
» lpBuffer$) As Long
01857
01858 Public Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName$, ByVal
» lpNewFileName$, ByVal bFailIfExists&) As Long
01859
01860 Public Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName$) As Long
01861
01862
01863 Public Const DELETE = &H10000
01864 Public Const READ_CONTROL = &H20000
01865 Public Const WRITE_DAC = &H40000
01866 Public Const WRITE_OWNER = &H80000
01867 Public Const SYNCHRONIZE As Long = &H100000
01868
01869 Public Const STANDARD_RIGHTS_READ = &H20000
01870 Public Const STANDARD_RIGHTS_WRITE = &H20000
01871 Public Const STANDARD_RIGHTS_EXECUTE = &H20000
01872 Public Const STANDARD_RIGHTS_REQUIRED = &HF0000
01873 Public Const STANDARD_RIGHTS_ALL = &H1F0000
01874
01875 ' Reg Key Security Options
01876 Public Const KEY_QUERY_VALUE = &H1
01877 Public Const KEY_SET_VALUE = &H2
01878 Public Const KEY_CREATE_SUB_KEY = &H4
01879 Public Const KEY_ENUMERATE_SUB_KEYS = &H8
01880 Public Const KEY_NOTIFY = &H10
01881 Public Const KEY_CREATE_LINK = &H20
01882 Public Const KEY_READ = ((STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY)
» And (Not SYNCHRONIZE))
01883 Public Const KEY_WRITE = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY) And (Not
» SYNCHRONIZE))
01884 Public Const KEY_EXECUTE = (KEY_READ)
01885 Public Const KEY_ALL_ACCESS = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE Or KEY_SET_VALUE Or
» KEY_CREATE_SUB_KEY Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY Or KEY_CREATE_LINK) And (Not SYNCHRONIZE))
01886
01887
01888 Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" (ByVal hKey&, ByVal lpSubKey$
» , ByVal ulOptions&, ByVal samDesired&, phkResult&) As Long
01889 Public Declare Function RegEnumKeyEx Lib "advapi32.dll" Alias "RegEnumKeyExA" (ByVal hKey&, ByVal dwIndex&,
» ByVal lpName$, lpcbName&, ByVal lpReserved&, ByVal lpClass$, lpcbClass&, lpftLastWriteTime As FILETIME)
» As Long
01890 Public Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey&) As Long
01891
01892
01893 ' GetDrivePublic Type return values
01894 Public Const DRIVE_REMOVABLE = 2
01895 Public Const DRIVE_FIXED = 3
01896 Public Const DRIVE_REMOTE = 4
01897 Public Const DRIVE_CDROM = 5
01898 Public Const DRIVE_RAMDISK = 6
01899
01900 Public Declare Function GetDriveType Lib "kernel32" Alias "GetDriveTypeA" (ByVal nDrive$) As Long
01901
01902 Public Declare Function InitiateSystemShutdown Lib "advapi32.dll" Alias "InitiateSystemShutdownA" (ByVal
» lpMachineName$, ByVal lpMessage$, ByVal dwTimeout&, ByVal bForceAppsClosed&, ByVal bRebootAfterShutdown&)
» As Long
01903

```

```

01904 Option Explicit
01905 ' demo project showing how to create a custom MSDE install
01906 ' by Bryan Stafford of New Vision Software - newvision@mmps.org
01907 ' this demo is released into the public domain "as is" without
01908 ' warranty or guaranty of any kind. In other words, use at
01909 ' your own risk.
01910 '
01911 ' Copyright © 2002 Bryan Stafford/New Vision Software
01912 '
01913 ' the code in this demo is copyrighted and may only be used in
01914 ' accordance with the rules set forth in the "House Rules"
01915 ' section of the web page found at http://www.mmps.org/vbvision/
01916 '
01917 ' see the general declararions section of MMain.bas for more info
01918 ' on this project
01919
01920
01921 Public Enum FILE_TIME_TYPE
01922     ftt_Created = 0&
01923     ftt_Accessed = 1&
01924     ftt_Modified = 2&
01925 End Enum
01926
01927 Public Enum COMPARE_FILE_VERSION_RETURN
01928     cfv_OneIsGreater = (-1&)
01929     cfv_Equal = 0&
01930     cfv_TwoIsGreater = 1&
01931 End Enum
01932
01933 Public Enum OSVERSION_RETURN
01934     [eNT3.51] = 1&
01935     eWin95 = 2&
01936     eWin98 = 3&
01937     eWinMe = 4&
01938     [eNT4.0] = 5&
01939     eWin2K = 6&
01940     eWinXP = 7&
01941     eNewerThanAll = 100&
01942 End Enum
01943
01944 Public Enum STRING_COMP_RETURN
01945     [A < B] = (-1&)
01946     [A = B] = 0&
01947     [A > B] = 1&
01948 End Enum
01949
01950 Private Type VS_FIXEDFILEINFO
01951     dwSignature As Long
01952     dwStrucVersionl As Integer
01953     dwStrucVersionh As Integer
01954     dwFileVersionMSI As Integer
01955     dwFileVersionMSh As Integer
01956     dwFileVersionLS As Long
01957     '     dwFileVersionLSI As Integer
01958     '     dwFileVersionLSh As Integer
01959     dwProductVersionMSI As Integer
01960     dwProductVersionMSh As Integer
01961     dwProductVersionLSI As Integer
01962     dwProductVersionLSh As Integer
01963     dwFileFlagsMask As Long
01964     dwFileFlags As Long
01965     dwFileOS As Long
01966     dwFileType As Long
01967     dwFileSubtype As Long
01968     dwFileDateMS As Long
01969     dwFileDateLS As Long
01970 End Type
01971
01972 Private Declare Function GetFileVersionInfo Lib "Version.dll" Alias "GetFileVersionInfoA" (ByVal
» IptstrFilename As String, ByVal dwHandle As Long, ByVal dwLen As Long, lpData As Any) As Long
01973 Private Declare Function GetFileVersionInfoSiZe Lib "Version.dll" Alias "GetFileVersionInfoSiZeA" (ByVal
» IptstrFilename As String, lpdwHandle As Long) As Long
01974 Private Declare Function VerQueryValue Lib "Version.dll" Alias "VerQueryValueA" (pBlock As Any, ByVal
» lpSubBlock As String, lp pBuffer As Any, puLen As Long) As Long
01975
01976 Public Function INDEXTOSTATEIMAGEMASK(ByVal iState As Long) As Long
01977     ' Rtns the one-based index of the state image shifted left twelve bits.
01978
01979 1 ' #define INDEXTOSTATEIMAGEMASK(i) ((i) << 12)

```

```

01980 1 INDEXTOSTATEIMAGEMASK = iState * (2 ^ 12)
01981 End Function
-----
01982
01983 Public Function Max(ByVal param1 As Long, ByVal param2 As Long) As Long
01984 ' Returns the larger of the two passed params
01985 If param1 > param2 Then Max = param1 Else Max = param2
01986 End Function
-----
01987
01988 Public Function Min(ByVal param1 As Long, ByVal param2 As Long) As Long
01989 ' Returns the smaller of the two passed params
01990 If param1 < param2 Then Min = param1 Else Min = param2
01991 End Function
-----
01992
01993 Public Function MAKELONG(ByVal wLow As Long, ByVal wHigh As Long) As Long
01994 ' Combines two integers into a long integer
01995 Dim nShiftHigh&
01996 CopyMemory ByVal VarPtr(nShiftHigh) + 2, ByVal VarPtr(wHigh), 2&
01997 MAKELONG = LOWORD(wLow) Or nShiftHigh
01998 End Function
-----
01999
02000 Public Function MAKELPARAM(ByVal wLow As Long, ByVal wHigh As Long) As Long
02001 ' Combines two integers into a long integer
02002 MAKELPARAM = MAKELONG(wLow, wHigh)
02003 End Function
-----
02004
02005 Public Function LOWORD(ByVal dwValue As Long) As Integer
02006 ' Returns the low 16-bit integer from a 32-bit long integer
02007 CopyMemory LOWORD, dwValue, 2&
02008 End Function
-----
02009
02010 Public Function HIWORD(ByVal dwValue As Long) As Integer
02011 ' Returns the high 16-bit integer from a 32-bit long integer
02012 CopyMemory HIWORD, ByVal VarPtr(dwValue) + 2, 2&
02013 End Function
-----
02014
02015 Public Function MAKEPOINT(ByVal x&, ByVal y&) As POINTAPI 'Double
02016 ' fills a POINTL struct
02017 MAKEPOINT.x = x
02018 MAKEPOINT.y = y
02019 End Function
-----
02020
02021 Public Function IsArrayEmpty(ByVal vArray As Variant) As Boolean
02022
02023 Dim lpSAFEARRAY&, nDims&
02024
02025 If (VarType(vArray) And vbArray) = vbArray Then
02026
02027 ' Get the pointer to the SAFEARRAY structure
02028 CopyMemory lpSAFEARRAY, ByVal VarPtr(vArray) + 8, 4
02029
02030 ' Get the number of dimensions
02031 CopyMemory nDims, ByVal lpSAFEARRAY, 2
02032
02033 ' The array is empty if doesn't have dimensions
02034 IsArrayEmpty = (nDims = 0)
02035
02036 Else
02037 Err.Raise 5
02038 End If
02039
02040 End Function
-----
02041
02042 Public Function StripNulls(ByVal sText As String) As String
02043 ' strips any nulls from the end of a string
02044 Dim nPosition&
02045
02046 StripNulls = sText
02047
02048 nPosition = InStr(sText, vbNullChar)
02049 If nPosition Then StripNulls = Left$(sText, nPosition - 1)
02050 If Len(sText) Then If Left$(sText, 1) = vbNullChar Then StripNulls = vbNullString
02051 End Function
-----
02052
02053 1 Public Function StripLfs(ByVal sText$) As String

```

```

1
02054
02055     Dim nLfLoc&
02056
02057     Do
02058         nLfLoc = InStr(sText, vbLf)
02059
02060         If nLfLoc Then sText = Left$(sText, nLfLoc - 1) & Right$(sText, Len(sText) - nLfLoc)
02061     Loop Until nLfLoc = 0
02062
02063     StripLfs = sText
02064
02065 End Function
-----
02066
02067 Public Function Replace(ByVal sText$, ByVal sToReplace$, ByVal sReplaceWith$) As String
02068
02069     Dim nRepLoc&
02070
02071     If sToReplace <> sReplaceWith Then
02072         Do
02073             nRepLoc = InStr(sText, sToReplace)
02074
02075             If nRepLoc Then sText = Left$(sText, nRepLoc - 1) & sReplaceWith & Right$(sText, Len(sText) - ((
»             nRepLoc - 1) + Len(sToReplace)))
02076         Loop Until nRepLoc = 0
02077     End If
02078
02079     Replace = sText
02080
02081 End Function
-----
02082
02083 Public Sub Yield()
02084     ' yields process time from calling routine
02085     Dim lpMsg As MSG
02086
02087     Do While PeekMessage(lpMsg, 0&, 0&, 0&, PM_REMOVE)
02088         Call TranslateMessage(lpMsg)
02089         Call DispatchMessage(lpMsg)
02090     Loop
02091
02092 End Sub
-----
02093
02094 Public Function ReturnModuleEXENAME() As String
02095     Dim sMePath$
02096
02097     sMePath = String$(MAX_PATH, 0)
02098
02099     Call GetModuleFileName(App.hInstance, sMePath, MAX_PATH)
02100
02101     ReturnModuleEXENAME = StripNulls(sMePath)
02102 End Function
-----
02103
02104 Public Function StringComp(ByVal A$, ByVal B$, ByVal eCompare As VbCompareMethod) As STRING_COMP_RETURN
02105     StringComp = StrComp(A, B, eCompare)
02106 End Function
-----
02107
02108 Public Function NumComp(ByVal number1 As Double, ByVal number2 As Double) As Long
02109     ' returns the same value that the VB StrComp function returns for strings except this function works on
»     numbers
02110     Select Case number1
02111     Case Is < number2: NumComp = (-1)
02112     Case Is = number2: NumComp = 0
02113     Case Is > number2: NumComp = 1
02114     End Select
02115 End Function
-----
02116
02117 Public Function CustomFormat(ByVal sValue$, ByVal sFormat$) As String
02118
02119     Dim sReturn$, nLenValue&, nLenFormat&, nLenReturn&
02120     Dim nTemp&, nRightOfValue&, nRightOfFormat&, nRightOfReturn&
02121
02122     sReturn = Format$(sValue, sFormat)
02123
02124     If InStr(sFormat, "#") Then ' optional chars present....
02125
02126         nTemp = InStr(sValue, ".")
02127         nLenValue = Len(sValue)
1 2

```

```

1 2
02128 If nTemp Then nRightOfValue = nLenValue - nTemp
02129
02130 nTemp = InStr(sFormat, ".")
02131 nLenFormat = Len(sFormat)
02132 If nTemp Then nRightOfFormat = nLenFormat - nTemp
02133
02134 nTemp = InStr(sReturn, ".")
02135 nLenReturn = Len(sReturn)
02136 If nTemp Then nRightOfReturn = nLenReturn - nTemp
02137
02138
02139 If (nRightOfReturn < nRightOfFormat) And (nRightOfValue > nRightOfReturn) Then
02140     If nRightOfValue <= nRightOfFormat Then
02141         CustomFormat = sReturn & String$(nRightOfValue - nRightOfReturn, vbKey0)
02142     Else
02143         CustomFormat = sReturn & String$(nRightOfFormat - nRightOfReturn, vbKey0)
02144     End If
02145
02146 Else
02147     CustomFormat = sReturn
02148 End If
02149 Else
02150     CustomFormat = sReturn
02151 End If
02152
02153 End Function

```

```

02154
02155 Public Sub TransparentBlt(hDestDC&, lpDestRect As RECT, hSrcDC&, lpSrcRect As RECT, TransColor&)
02156
02157     Dim hInvDC&, hMaskDC&, hResultDC&, hInvBmp&, hMaskBmp&, hResultBmp&
02158     Dim hInvPrevBmp&, hMaskPrevBmp&, hDestPrevBmp&, nSrcWidth&, nSrcHeight&, nOriginalColor&
02159
02160     With lpSrcRect
02161         nSrcWidth = .Right - .Left
02162         nSrcHeight = .Bottom - .Top
02163     End With
02164
02165     ' create the mask and invert stage DCs and bitmaps
02166     hInvDC = CreateCompatibleDC(hDestDC)
02167     hMaskDC = CreateCompatibleDC(hDestDC)
02168     ' monochrome bitmaps for the masks
02169     hInvBmp = CreateBitmap(nSrcWidth, nSrcHeight, 1, 1, ByVal 0&)
02170     hMaskBmp = CreateBitmap(nSrcWidth, nSrcHeight, 1, 1, ByVal 0&)
02171     hInvPrevBmp = SelectObject(hInvDC, hInvBmp)
02172     hMaskPrevBmp = SelectObject(hMaskDC, hMaskBmp)
02173
02174     ' create the DC and bitmap to hold the result
02175     hResultDC = CreateCompatibleDC(hDestDC)
02176     ' color bitmap for final result
02177     hResultBmp = CreateCompatibleBitmap(hDestDC, nSrcWidth, nSrcHeight)
02178     hDestPrevBmp = SelectObject(hResultDC, hResultBmp)
02179
02180     ' create mask: set background color of source to transparent color.
02181     nOriginalColor = SetBkColor(hSrcDC, TransColor)
02182     With lpSrcRect
02183         Call BitBlt(hMaskDC, 0, 0, nSrcWidth, nSrcHeight, hSrcDC, .Left, .Top, vbSrcCopy)
02184     End With
02185     TransColor = SetBkColor(hSrcDC, nOriginalColor)
02186
02187     ' create inverse of mask to AND w/ source & combine w/ background.
02188     Call BitBlt(hInvDC, 0, 0, nSrcWidth, nSrcHeight, hMaskDC, 0, 0, vbNotSrcCopy)
02189
02190     ' copy background bitmap to result & create final transparent bitmap
02191     With lpDestRect
02192         Call BitBlt(hResultDC, 0, 0, nSrcWidth, nSrcHeight, hDestDC, .Left, .Top, vbSrcCopy)
02193
02194         ' AND mask bitmap w/ result DC to punch hole in the background by
02195         ' painting black area for non-transparent portion of source bitmap.
02196         Call BitBlt(hResultDC, 0, 0, nSrcWidth, nSrcHeight, hMaskDC, 0, 0, vbSrcAnd)
02197
02198         ' AND inverse mask w/ source bitmap to turn off bits associated
02199         ' with transparent area of source bitmap by making it black.
02200         Call BitBlt(hSrcDC, 0, 0, nSrcWidth, nSrcHeight, hInvDC, 0, 0, vbSrcAnd)
02201
02202         ' XOR result w/ source bitmap to make background show through.
02203         Call BitBlt(hResultDC, 0, 0, nSrcWidth, nSrcHeight, hSrcDC, 0, 0, vbSrcPaint)
02204
02205         ' copy result to the dest DC
02206     End With
02206 1 2 Call BitBlt(hDestDC, .Left, .Top, nSrcWidth, nSrcHeight, hResultDC, 0, 0, vbSrcCopy)

```

```

02207 1 2 End With
02208
02209 ' clean up after ourselves
02210 Call DeleteObject(SelectObject(hMaskDC, hMaskPrevBmp))
02211 Call DeleteObject(SelectObject(hI nvDC, hI nvPrevBmp))
02212 Call DeleteObject(SelectObject(hResul tDC, hDestPrevBmp))
02213
02214 DeleteDC hMaskDC
02215 DeleteDC hI nvDC
02216 DeleteDC hResul tDC
02217
02218 End Sub

```

```

02219
02220 Public Function GetOSVersion() As OSVERSION_RETURN
02221 ' Returns the version of the OS that we are running on
02222 Dim osInfo As OSVERSIONINFO
02223
02224 osInfo.dwOSVersionInfoSize = 148&
02225 Call GetVersionEx(osInfo)
02226
02227 With osInfo
02228     Select Case .dwPlatformId
02229     Case 1
02230         If .dwMinorVersion = 0 Then ' <-- Win95
02231             GetOSVersion = eWin95
02232         ElseIf .dwMinorVersion = 10 Then ' <-- Win98
02233             GetOSVersion = eWin98
02234         ElseIf .dwMinorVersion = 90 Then ' <-- WinME
02235             GetOSVersion = eWinME
02236         End If
02237     Case 2
02238         If .dwMajorVersion = 3 Then ' <-- NT 3.51
02239             GetOSVersion = [eNT3.51]
02240         ElseIf .dwMajorVersion = 4 Then ' <-- NT 4.0
02241             GetOSVersion = [eNT4.0]
02242         ElseIf (.dwMajorVersion = 5) And _
02243             (.dwMinorVersion = 0) Then
02244             ' <-- Win2K
02245             GetOSVersion = eWin2K
02246         ElseIf (.dwMajorVersion = 5) And _
02247             (.dwMinorVersion = 1) Then
02248             ' <-- WinXP
02249             GetOSVersion = eWinXP
02250         ElseIf (.dwMajorVersion > 5) Or _
02251             (.dwMinorVersion > 1) Then
02252             GetOSVersion = eNewerThanAll
02253         End If
02254     Case Else
02255         GetOSVersion = True
02256     End Select
02257 End With
02258
02259 End Function

```

```

02262
02263 Public Function GetFileVersion(strFilename As String) As String
02264
02265 Dim lngTmp&, abyBuf() As Byte, lngBufLen&, lngVerPointer&
02266 Dim VerBuffer As VS_FIXEDFILEINFO, lVerbufferLen&, strTmp$
02267
02268 ' Initialize the return value
02269 strTmp = ""
02270
02271 ' Get the size needed for the buffer
02272 lngBufLen = GetFileVersionInfoSize(strFilename, lngTmp)
02273
02274 If lngBufLen >= 1 Then
02275     ' Call succeeded, grow the array
02276     ReDim abyBuf(lngBufLen) As Byte
02277
02278     ' Get the version information
02279     Call GetFileVersionInfo(strFilename, 0&, lngBufLen, abyBuf(0))
02280     Call VerQueryValue(abyBuf(0), "\", lngVerPointer, lVerbufferLen)
02281     CopyMemory ByVal VarPtr(VerBuffer), ByVal lngVerPointer, Len(VerBuffer)
02282
02283     ' Get the constituent parts into the string
02284     1 2 3 With VerBuffer

```

```

02285 1 2 3 strTmp = Format$(.dwFileVersionMS) & "." & Format$(.dwFileVersionMSI, "00") & "." & Format$(.
»      dwFileVersionLS, "0000")
02286      End With
02287      End If
02288
02289      GetFileVersion = strTmp
02290
02291  End Function

```

```

02292
02293  Public Function CompareFileVersions(ByVal sVersionOne$, sVersionTwo$) As COMPARE_FILE_VERSION_RETURN
02294
02295      Dim i&, nLenOne&, nLenTwo&
02296
02297      nLenOne = Len(sVersionOne)
02298      nLenTwo = Len(sVersionTwo)
02299
02300      Select Case True
02301      Case nLenOne > nLenTwo: sVersionTwo = String$(nLenOne - nLenTwo, "0") & sVersionTwo
02302      Case nLenOne < nLenTwo
02303          sVersionOne = String$(nLenTwo - nLenOne, "0") & sVersionOne
02304          nLenOne = nLenTwo
02305      End Select
02306
02307      For i = 1 To nLenOne
02308          Select Case True
02309          Case Val(Mid$(sVersionOne, i, 1)) > Val(Mid$(sVersionTwo, i, 1))
02310              CompareFileVersions = cfv_OneIsGreater
02311              Exit For
02312
02313          Case Val(Mid$(sVersionOne, i, 1)) < Val(Mid$(sVersionTwo, i, 1))
02314              CompareFileVersions = cfv_TwoIsGreater
02315              Exit For
02316
02317          End Select
02318      Next
02319
02320  End Function

```

```

02321
02322  Public Function ReturnPathRoot(ByVal sPath$) As String
02323
02324      Dim i&
02325
02326      i = InStr(sPath, ":")
02327
02328      If i Then
02329          ReturnPathRoot = Left$(sPath, i + 1)
02330      Else
02331          i = InStr(sPath, "\\")
02332
02333          If i Then
02334              If Len(sPath) > i + 2 Then
02335                  i = InStr(i + 2, sPath, "\")
02336
02337                  If i Then
02338                      If Len(sPath) > i + 1 Then
02339                          i = InStr(i + 1, sPath, "\")
02340
02341                          If i Then ReturnPathRoot = Left$(sPath, i)
02342                      End If
02343                  End If
02344              End If
02345          End If
02346      End If
02347
02348  End Function

```

```

02349
02350  Public Function ReturnFileName(ByVal sFilePath$, Optional ByRef sPathString$) As String
02351      ' returns the file name from a path as the return value and the rest of the
02352      ' path in the sPathString param.
02353
02354      Dim nLastPathSep&, lpString&, nStrLen&, nFileNameLen&, sReturn$
02355
02356      lpString = StrPtr(sFilePath)
02357      nStrLen = LenB(sFilePath)
02358
02359      If lpString Then nLastPathSep = Callwcsrchr(lpString, 92) ' <-- the "\ char
02360
02361  1 2 If nLastPathSep Then

```

```

02362 1 2 nLastPathSep = nLastPathSep + 2
02363
02364 nFileNameLen = (lpString + nStrLen) - nLastPathSep
02365
02366 ' get the folder...
02367 sPathString = Left$(sFilePath, Len(sFilePath) - ((nFileNameLen \ 2) + 1))
02368 If Right$(sPathString, 1) = ":" Then sPathString = sPathString & "\"
02369
02370 sReturn = String$(nFileNameLen \ 2, vbNullChar)
02371
02372 CopyMemory ByVal StrPtr(sReturn), ByVal nLastPathSep, nFileNameLen
02373
02374 ReturnFileName = sReturn
02375 End If
02376
02377 End Function

```

```

02378
02379 Public Function ReturnFileExtension(ByVal sFileName$, Optional ByRef sFileString$) As String
02380
02381 If sFileName <> vbNullString Then
02382 Dim nLastExtensionSep, lpString, nStrLen, nFileNameLen, sReturn$
02383
02384 lpString = StrPtr(sFileName)
02385 nStrLen = LenB(sFileName)
02386
02387 If lpString Then nLastExtensionSep = CallWcsrchr(lpString, 46) '<-- the "." char
02388
02389 If nLastExtensionSep Then
02390 nFileNameLen = (lpString + nStrLen) - nLastExtensionSep
02391
02392 sFileString = Left$(sFileName, Len(sFileName) - (nFileNameLen \ 2))
02393
02394 sReturn = String$(nFileNameLen \ 2, vbNullChar)
02395
02396 CopyMemory ByVal StrPtr(sReturn), ByVal nLastExtensionSep, nFileNameLen
02397
02398 ReturnFileExtension = sReturn
02399 End If
02400 End If
02401
02402 End Function

```

```

02403
02404 Public Function CallWcsrchr(ByVal lpString, ByVal nChar) As Long
02405
02406 Dim i, nLen, alnt() As Integer
02407
02408 nLen = lstrlenW(lpString) + 1 '<-- account for the null terminator
02409
02410 If nLen Then
02411 ReDim alnt(nLen - 1) As Integer
02412
02413 Call CopyMemory(alnt(0), ByVal lpString, nLen * 2)
02414
02415 For i = (nLen - 1) To 0 Step -1
02416 If alnt(i) = nChar Then
02417 CallWcsrchr = lpString + (i * 2)
02418
02419 Exit For
02420 End If
02421 Next
02422 End If
02423
02424 End Function

```

```

02425
02426 Public Function RemoveFileNameFromPath(ByVal sFilePath$) As String
02427
02428 Dim sPathString$
02429
02430 Call ReturnFileName(sFilePath, sPathString)
02431
02432 RemoveFileNameFromPath = sPathString
02433
02434 End Function

```

```

02435
02436 Public Function ReturnShortPathName(ByVal sLongPath$) As String
02437
02438 Dim sBuffer$, sReturn$

```

```

02439 1
02440     sBuffer = String$(MAX_PATH, 0)
02441
02442     If GetShortPathName(sLongPath, sBuffer, MAX_PATH) Then
02443         sReturn = StripNulls(sBuffer)
02444     Else
02445         sReturn = sLongPath
02446     End If
02447
02448     ReturnShortPathName = sReturn
02449
02450 End Function
02451


---


02452
02453 Public Function AddBackslashToPath(ByVal sPath$) As String
02454
02455     If Len(sPath) Then If Right$(sPath, 1) <> "\" Then sPath = sPath & "\"
02456
02457     AddBackslashToPath = sPath
02458
02459 End Function
02460


---


02461
02462 Public Function ReturnComputerName() As String
02463
02464     Dim sBuffer$, sReturn$
02465
02466     sBuffer = String$(30, 0)
02467
02468     If GetComputerName(sBuffer, 30) Then
02469         ReturnComputerName = StripNulls(sBuffer)
02470     End If
02471
02472 End Function
02473


---


02474
02475 Public Function TrueFalseToYesNo(ByVal bValue As Boolean) As String
02476
02477     If bValue Then
02478         TrueFalseToYesNo = "Yes"
02479     Else
02480         TrueFalseToYesNo = "No"
02481     End If
02482
02483 End Function
02484


---


02485
02486 Public Function FileExists(ByVal sFileName$) As Boolean
02487
02488     'checks if a file or dir exists
02489     'returns true if it does, returns false otherwise
02490     Dim hFile&, Win32FindData As WIN32_FIND_DATA_A
02491
02492     sFileName = Trim$(sFileName)
02493     hFile = FindFirstFile(sFileName, Win32FindData)
02494
02495     If (hFile <> INVALID_HANDLE_VALUE) And (hFile <> ERROR_NO_MORE_FILES) Then
02496         FileExists = True
02497     ElseIf GetFileAttributes(sFileName) <> (-1) Then
02498         ' FindFirstFile will not return the root dir of a drive so we check the attributes
02499         ' of sFileName in case it is the root
02500         FileExists = True
02501     End If
02502
02503     Call FindClose(hFile)
02504
02505 End Function
02506


---


02507
02508 Private Function DriveExists(ByVal sDrvName$) As Boolean
02509
02510     Dim sAllDrives$
02511
02512     'pad the string with spaces
02513     sAllDrives = String$(MAX_PATH, 0)
02514
02515     'call the API to get the string containing all drives
02516     Call GetLogicalDriveStrings(MAX_PATH, sAllDrives)
02517
02518     'Do an InStr search, using text compare (not case sensitive).
02519     'If the drive letter is in the string, InStr will return
02520     'greater than 0, evaluating below as True
02521     DriveExists = InStr(1, sAllDrives, sDrvName, vbTextCompare) > 0
02522
02523 End Function

```

```

02516
02517 Public Function CreateTempFileName(ByVal sPath$, ByVal sPrefix$, ByVal sSuffix$) As String
02518
02519     Dim i&, sTemp$
02520
02521     sPath = AddBackslashToPath(sPath)
02522
02523     Do
02524         i = i + 1
02525
02526         sTemp = sPrefix & CStr(i) & sSuffix
02527     Loop Until FileExists(sPath & sTemp) = False
02528
02529     CreateTempFileName = sPath & sTemp
02530
02531 End Function

```

```

02532
02533 Public Function ReturnProcAddress(ByVal lpProc&) As Long
02534     ReturnProcAddress = lpProc
02535 End Function

```

```

02536
02537 Public Function IsCallExported(ByVal ModuleName As String, ByVal ProcName As String) As Boolean
02538     Dim hModule As Long
02539     Dim lpProc As Long
02540     Dim FreeLib As Boolean
02541
02542     ' check first to see if the module is already
02543     ' mapped into this process.
02544     hModule = GetModuleHandle(ModuleName)
02545     If hModule = 0 Then
02546         ' need to load module into this process.
02547         hModule = LoadLibrary(ModuleName)
02548         FreeLib = True
02549     End If
02550
02551     ' if the module is mapped, check procedure
02552     ' address to verify it's exported.
02553     If hModule Then
02554         lpProc = GetProcAddress(hModule, ProcName)
02555         IsCallExported = (lpProc <> 0)
02556     End If
02557
02558     ' unload library if we loaded it here.
02559     If FreeLib Then Call FreeLibrary(hModule)
02560 End Function

```

```

02561
02562 Public Function ReturnFileTime(ByVal sFile$, eReturnType As FILE_TIME_TYPE, bConvertToLocalTime As Boolean)
As Date
02563     »
02564     Dim ftFTCreate As FILETIME, ftFTAccess As FILETIME, ftFTWrite As FILETIME
02565     Dim hFile&
02566
02567     ' Get a handle to the file
02568     hFile = CreateFile(sFile, GENERIC_READ Or GENERIC_WRITE, FILE_SHARE_READ Or FILE_SHARE_WRITE, ByVal 0&, _
02569
02570         OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL Or FILE_FLAG_RANDOM_ACCESS, ByVal 0&)
02571
02572     ' Make sure we got a valid handle
02573     If hFile <> INVALID_HANDLE_VALUE Then
02574         If GetFileTime(hFile, ftFTCreate, ftFTAccess, ftFTWrite) Then
02575             ' Convert the Windows time format to something we can use
02576             Select Case eReturnType
02577                 Case ftt_Accessed: ReturnFileTime = WinTimeToVBATime(ftFTAccess, bConvertToLocalTime)
02578                 Case ftt_Modified: ReturnFileTime = WinTimeToVBATime(ftFTWrite, bConvertToLocalTime)
02579                 Case ftt_Created: ReturnFileTime = WinTimeToVBATime(ftFTCreate, bConvertToLocalTime)
02580             End Select
02581         End If
02582
02583         Call CloseHandle(hFile)
02584     End If
02585
02586 End Function

```

```

02588
02589 Public Function WinTimeToVBATime(ftWinFileTime As FILETIME, bConvertToLocalTime As Boolean) As Date
02590

```

```

02591 1 Dim stSystemFileTime As SYSTEMTIME, ftLocalFileTime As FILETIME
02592
02593 ' Was local time requested?
02594 If bConvertToLocalTime Then
02595     ' Need to convert the file time to a local time
02596     Call FileTimeToLocalFileTime(ftWinFileTime, ftLocalFileTime)
02597     ftWinFileTime = ftLocalFileTime
02598 End If
02599
02600 ' Call the API to convert the time format
02601 If FileTimeToSystemTime(ftWinFileTime, stSystemFileTime) Then
02602     With stSystemFileTime
02603         WinTimeToVBATime = DateSerial(.wYear, .wMonth, .wDay) + TimeSerial(.wHour, .wMinute, .wSecond)
02604     End With
02605 End If
02606
02607 End Function

```

```

02608
02609 Public Function RemoveIllegalFileChars(ByVal sFileName$) As String
02610
02611     sFileName = Replace(sFileName, "\", "_")
02612     sFileName = Replace(sFileName, "/", "_")
02613     sFileName = Replace(sFileName, ":", "_")
02614     sFileName = Replace(sFileName, "*", "_")
02615     sFileName = Replace(sFileName, "?", "_")
02616     sFileName = Replace(sFileName, Chr$(34), "_")
02617     sFileName = Replace(sFileName, "<", "_")
02618     sFileName = Replace(sFileName, ">", "_")
02619     sFileName = Replace(sFileName, "|", "_")
02620
02621     RemoveIllegalFileChars = sFileName
02622
02623 End Function

```

```

02624
02625 Public Function Is2KShell() As Boolean
02626     ' this function returns the version of the Comdlg32.dll on the system
02627     ' this info is used to determine which version of the OPENFILENAME struct
02628     ' should be passed to the dialog functions
02629
02630     Dim nBufferSize&, nDiscard&, lpBuffer&, nVerMajor&, abyBuffer() As Byte
02631
02632     Const FILE_NAME As String = "Comdlg32.dll"
02633
02634     nBufferSize = GetFileVersionInfoSize(FILE_NAME, nDiscard)
02635
02636     If nBufferSize > 0 Then
02637         ReDim abyBuffer(nBufferSize - 1) As Byte
02638
02639         Call GetFileVersionInfo(FILE_NAME, 0&, nBufferSize, abyBuffer(0))
02640
02641         If VerQueryValue(abyBuffer(0), "\", lpBuffer, nDiscard) Then
02642             CopyMemory nVerMajor, ByVal lpBuffer + 10, 2&
02643
02644             If nVerMajor >= 5 Then Is2KShell = True
02645         End If
02646     End If
02647
02648 End Function

```

```

02649
02650 Public Function PointerToStringA(ByVal lpStringA As Long) As String
02651     Dim Buffer() As Byte
02652     Dim nLen As Long
02653
02654     If lpStringA Then
02655         nLen = IStrLenA(ByVal lpStringA)
02656         If nLen Then
02657             ReDim Buffer(0 To (nLen - 1)) As Byte
02658             CopyMemory Buffer(0), ByVal lpStringA, nLen
02659             PointerToStringA = StrConv(Buffer, vbUnicode)
02660         End If
02661     End If
02662 End Function

```

```

02663
02664 Public Function PointerToStringW(ByVal lpStringW As Long) As String
02665     Dim Buffer() As Byte
02666     Dim nLen As Long
02667

```

```
02668 1  If IpStringW Then
02669     nLen = IStrLenW(IpStringW) * 2
02670     If nLen Then
02671         ReDim Buffer(0 To (nLen - 1)) As Byte
02672         CopyMemory Buffer(0), ByVal IpStringW, nLen
02673         PointerToStringW = Buffer
02674     End If
02675 End If
02676 End Function

02677
02678 Public Function PointerToDWord(ByVal IpDWord As Long) As Long
02679     Dim nRet As Long
02680     If IpDWord Then
02681         CopyMemory nRet, ByVal IpDWord, 4
02682         PointerToDWord = nRet
02683     End If
02684 End Function
02685
02686
```

```

02687 Option Explicit
02688 ' demo project showing how to create a custom MSDE install
02689 ' by Bryan Stafford of New Vision Software - newvision@mvps.org
02690 ' this demo is released into the public domain "as is" without
02691 ' warranty or guaranty of any kind. In other words, use at
02692 ' your own risk.
02693 '
02694 ' Copyright © 2002 Bryan Stafford/New Vision Software
02695 '
02696 ' the code in this demo is copyrighted and may only be used in
02697 ' accordance with the rules set forth in the "House Rules"
02698 ' section of the web page found at http://www.mvps.org/vbvision/
02699 '
02700 ' see the general declararions section of MMain.bas for more info
02701 ' on this project

```

```

02702
02703
02704 Public Function AskAboutCancel() As Boolean
02705     If MsgBox("This action will cancel any remaining operations." _
02706         & vbNewLine & vbNewLine & "Are you suer you wish to cancel?", vbYesNo Or vbQuestion, nvMSGBOXCAPTION)
02707         = vbYes Then
02708             AskAboutCancel = True
02709     End If
02710 End Function

```

```

02711
02712 Public Function CreateTitlePicture(ByVal hWndTarget&) As StdPicture
02713     ' center the foreground bitmap onto the existing background
02714
02715     Dim hOwnerDC&, hForegroundDC&, hBackgroundDC&, hStretchDC&
02716     Dim hBmpObj&, ptForegroundSize As POINTAPI, ptBkgndSize As POINTAPI, Bmp As BITMAP
02717     Dim nClientWidth&, nClientHeight&, rectStretch As RECT, rectForeground As RECT
02718     Dim hBackgroundBmp&, hPrevBackgroundBmp&, hForegroundBmp&, hPrevForegroundBmp&, hPrevStretchBmp&
02719
02720     ' create the DCs we will use
02721     hOwnerDC = GetDC(hWndTarget)
02722     hForegroundDC = CreateCompatibleDC(hOwnerDC)
02723     hBackgroundDC = CreateCompatibleDC(hOwnerDC)
02724     SaveDC hForegroundDC
02725     SaveDC hBackgroundDC
02726
02727
02728     Dim hModule&
02729
02730     ' for this demo, we get the resource from the exe when running in the IDE.
02731     ' this is because the resource is not compiled into the module pointed to
02732     ' by App.hInstance when this code is run in the IDE
02733     If RunnngInIde() Then
02734         Dim sPath$
02735         sPath = AddBackslashToPath(App.Path)
02736
02737         hModule = LoadLibrary(sPath & PROJECT_EXE_NAME)
02738     Else
02739         hModule = App.hInstance
02740     End If
02741
02742     If hModule = 0 Then MsgBox "Unable to find resources!", vbExclamation
02743
02744     ' load the foreground bitmap
02745     hForegroundBmp = LoadBitmapByNum(hModule, 200&)
02746
02747     ' get the bitmap info so that we can get the size of the bitmap
02748     If GetObjectAPI(hForegroundBmp, Len(Bmp), Bmp) Then
02749         With ptForegroundSize
02750             .x = Bmp.bmWidth
02751             .y = Bmp.bmHeight
02752         End With
02753     Else
02754         MsgBox "Unable to retrieve bitmap info in ""CenterForeground""."
02755     End If
02756
02757     ' save the previous bitmap when we select the new one into the DC
02758     hPrevForegroundBmp = SelectObject(hForegroundDC, hForegroundBmp)
02759
02760
02761     ' free the library if we are running in the IDE
02762     If RunnngInIde() Then
02763

```

```

02764 1 2 Call FreeLibrary(hModule)
02765 End If
02766
02767 ' get the rect of the window
02768 Call GetClientRect(hWndTarget, rectStretch)
02769
02770
02771 With rectStretch
02772 ' store the client dimentions for later use
02773 nClientWidth = .Right
02774 nClientHeight = .Bottom
02775
02776 ' create a new bitmap and select it into the stretch DC
02777 hPrevBackgroundBmp = SelectObject(hBackgroundDC, CreateCompatibleBitmap(hOwnerDC, .Right, .Bottom))
02778
02779
02780 ' copy the existing background bitmap onto the stretched DC
02781 Call FillRect(hBackgroundDC, rectStretch, (COLOR_BTNFACE + 1))
02782
02783 ' release the owner DC
02784 ReleaseDC hWndTarget, hOwnerDC
02785 ' set the dimentions of the graphic
02786 With rectForeground
02787 .Right = ptForegroundSize.x
02788 .Bottom = ptForegroundSize.y
02789 End With
02790
02791 ' calculate the location for the foreground bitmap
02792 .Left = (.Right \ 2) - (ptForegroundSize.x \ 2)
02793 .Right = .Left + ptForegroundSize.x
02794 .Top = 0 ' (.bottom \ 2) - (ptForegroundSize.y \ 2)
02795 .Bottom = .Top + ptForegroundSize.y
02796
02797 ' copy the graphic onto the background using the transparent color
02798 TransparentBlt hBackgroundDC, rectStretch, hForegroundDC, rectForeground, vbWhite
02799
02800 End With
02801
02802 ' create a picture object and return it to the caller
02803 Set CreateTitlePicture = PictureFromDC(hBackgroundDC, 0, 0, nClientWidth, nClientHeight)
02804
02805
02806 ' delete the memory DCs used for holding the bitmaps
02807 DeleteObject SelectObject(hForegroundDC, hPrevForegroundBmp)
02808 RestoreDC hForegroundDC, (-1&)
02809 DeleteDC hForegroundDC
02810
02811 DeleteObject SelectObject(hBackgroundDC, hPrevBackgroundBmp)
02812 RestoreDC hBackgroundDC, (-1&)
02813 DeleteDC hBackgroundDC
02814
02815 End Function

```

```

02816
02817 Private Function PictureFromDC(ByVal hDCSrc&, ByVal nLeft&, ByVal nTop&, ByVal nWidth&, ByVal nHeight&) As
» StdPicture
02818 Dim hDCMemory&, hBmp&, hBmpPrev&, hPal&, hPalPrev&
02820 Dim fHasPalette&, nPaletteEntries&, LogPal As LOGPALETTE256
02821
02822 ' create the DC and bitmap we will use
02823 hDCMemory = CreateCompatibleDC(hDCSrc)
02824 hBmp = CreateCompatibleBitmap(hDCSrc, nWidth, nHeight)
02825 hBmpPrev = SelectObject(hDCMemory, hBmp)
02826
02827 ' determine whether or not the video supports 256 color palettes
02828 nPaletteEntries = GetDeviceCaps(hDCSrc, SI_ZPALETTE)
02829 fHasPalette = GetDeviceCaps(hDCSrc, RASTERCAPS) And RC_PALETTE
02830
02831 ' if this is 256 color video, we need to create and add a palette to our DC
02832 If fHasPalette And (nPaletteEntries = 256) Then
02833 LogPal.palVersion = &H300
02834 LogPal.palNumEntries = 256
02835 GetSystemPaletteEntries hDCSrc, 0, 256, LogPal.palPalEntry(0)
02836 hPal = CreatePalette(LogPal)
02837 hPalPrev = SelectPalette(hDCMemory, hPal, 0)
02838 RealizePalette hDCMemory
02839 End If
02840
02841 1 ' copy the passed image into the local DC

```

```

02842 1 | BitBlt hDCMemory, 0, 0, nWidth, nHeight, hDCSrc, nLeft, nTop, vbSrcCopy
02843 |
02844 | ' get the bitmap from the DC
02845 | hBmp = SelectObject(hDCMemory, hBmpPrev)
02846 |
02847 | ' if we created a palette, get it from the DC
02848 | If fHasPalette And (nPaletteEntries = 256) Then
02849 |     hPal = SelectPalette(hDCMemory, hPalPrev, 0)
02850 | End If
02851 |
02852 | ' clean up
02853 | DeleteDC hDCMemory
02854 |
02855 | ' create a picture from the bitmap and return it to the caller
02856 | Set PictureFromDC = PictureFromBitmap(hBmp, hPal)
02857 |
02858 | End Function

```

```

02859 |
02860 | Private Function PictureFromBitmap(ByVal hBmp&, ByVal hPal&) As StdPicture
02861 |
02862 |     Dim IPictureIID As GUID, IPic As IPicture, tagPic As PICTDESC_BMP
02863 |
02864 |     ' fill in the IPicture GUID {7BF80980-BF32-101A-8BBB-00AA00300CAB}
02865 |     With IPictureIID
02866 |         .Data1 = &H7BF80980
02867 |         .Data2 = &HBF32
02868 |         .Data3 = &H101A
02869 |         .Data4(0) = &H8B
02870 |         .Data4(1) = &HBB
02871 |         .Data4(2) = &HO
02872 |         .Data4(3) = &HAA
02873 |         .Data4(4) = &HO
02874 |         .Data4(5) = &H30
02875 |         .Data4(6) = &HC
02876 |         .Data4(7) = &HAB
02877 |     End With
02878 |
02879 |     ' set the properties on the picture object
02880 |     With tagPic
02881 |         .Size = Len(tagPic)
02882 |         .Type = vbPicTypeBitmap
02883 |         .hBmp = hBmp
02884 |         .hPal = hPal
02885 |     End With
02886 |
02887 |     ' create a picture that will delete its bitmap when it is finished with it
02888 |     OleCreatePictureIndirect tagPic, IPictureIID, API_TRUE, IPic
02889 |
02890 |     ' return the picture to the caller
02891 |     Set PictureFromBitmap = IPic
02892 |
02893 | End Function

```

```

02894 |
02895 | Public Function RunningInIde() As Boolean
02896 |     ' This function will return False if running
02897 |     ' compiled code, True if running IDE
02898 |     Debug.Assert Not TestIDE(RunningInIde)
02899 | End Function

```

```

02900 |
02901 | Private Function TestIDE(test As Boolean) As Boolean
02902 |     test = True
02903 | End Function
02904 |

```

```

02905 Option Explicit
02906 ' demo project showing how to create a custom MSDE install
02907 ' by Bryan Stafford of New Vision Software - newvision@mmps.org
02908 ' this demo is released into the public domain "as is" without
02909 ' warranty or guaranty of any kind. In other words, use at
02910 ' your own risk.
02911 '
02912 ' Copyright © 2002 Bryan Stafford/New Vision Software
02913 '
02914 ' the code in this demo is copyrighted and may only be used in
02915 ' accordance with the rules set forth in the "House Rules"
02916 ' section of the web page found at http://www.mmps.org/vbvision/
02917 '
02918 '
02919 ' The MS SQL Server team has changed their mind about the suggested installation
02920 ' methods for MSDE. This demo has been altered from the previous version to
02921 ' support the currently supported installation method.
02922 '
02923 ' Instructions for generating a custom Desktop Engine installation....
02924 '
02925 ' 1.) Copy the all 16 of the SQLRUNxx.MSI files and the SQLRUN.CAB file from the MSDE
02926 ' install (use the latest SP install) to the root directory of your installation project.
02927 '
02928 '
02929 ' Further instructions for using this installation utility are as follows....
02930 '
02931 ' 1.) Customize the information in the constants below to reflect your unique
02932 ' installation info
02933 '
02934 '
02935 ' 2.) The following are the files that would be distributed with this install:
02936 '
02937 ' - The "AUTORUN.INF" file to auto-run a CD-ROM install
02938 '
02939 ' - The installer app itself "MSDE_Installer.exe"
02940 '
02941 ' - All sixteen of the SQLRUNxx.MSI files
02942 '
02943 ' - MSI directory containing the ANSI and UNICODE versions of the MS Installer
02944 ' setup ("INSTMSI.EXE" and "INSTMSIW.EXE") from the MSDE CD.
02945 '
02946 ' - The CAB file ("SQLRUN.CAB") that corresponds to the MSI files you are using for
02947 ' your install. You should use the files from the latest MSDE SP
02948 '
02949 ' - The "nvMSI_ERR.dll" library that is used to return extended error info from
02950 ' the MSDE install.
02951 '
02952 ' The structure would look like the following....
02953 '
02954 ' Root of the install CD:
02955 ' AUTORUN.INF
02956 ' SQLRUN01.MSI
02957 ' SQLRUN02.MSI
02958 ' SQLRUN03.MSI
02959 ' SQLRUN04.MSI
02960 ' SQLRUN05.MSI
02961 ' SQLRUN06.MSI
02962 ' SQLRUN07.MSI
02963 ' SQLRUN08.MSI
02964 ' SQLRUN09.MSI
02965 ' SQLRUN10.MSI
02966 ' SQLRUN11.MSI
02967 ' SQLRUN12.MSI
02968 ' SQLRUN13.MSI
02969 ' SQLRUN14.MSI
02970 ' SQLRUN15.MSI
02971 ' SQLRUN16.MSI
02972 ' MSDE_Installer.exe
02973 ' nvMSI_ERR.dll
02974 ' SQLRUN.CAB
02975 '
02976 ' Sub-directory named "MSI" containing the following files:
02977 ' INSTMSI.EXE
02978 ' INSTMSIW.EXE
02979 '
02980 '
02981 '
02982 ' To change the logo for the project, create a logo about the same size as the
02983 ' the one in the logo bitmap in the project directory. then replace the current

```

```

02984 ' bitmap in the resource file in the project using the resource editor
02985 '
02986
02987
02988 ' used in the registry keys for running the configuration when a re-boot is required
02989 Public Const COMPANY_NAME As String = "Your_Company_Name"
02990
02991 ' it is *always* a good idea to set a password for the default 'sa' login
02992 Private Const SA_PASSWORD As String = "admin"
02993
02994
02995 ' the default info for this instance of MSDE that will be displayed to the user when the
02996 ' server data dialog is shown for the first time
02997 Private Const DEFAULT_SERVER_INSTANCE_NAME As String = "YOURINSTANCE" '<- if you want to install a
» non-named instance of MSDE enter "DEFAULT" for the instance name
02998 Private Const DEFAULT_USER_ID As String = "USERID"
02999 Private Const DEFAULT_PASSWORD As String = "abc123"
03000
03001 ' the name of the compiled exe from this project. this is used to retrieve the logo bitmap while running
» in the IDE
03002 Public Const PROJECT_EXE_NAME As String = "MSDE_Installer.exe"
03003
03004
03005
03006
03007 Public Enum NAVIGATE_FLAGS
03008     eNext = 1&
03009     eBack = 2&
03010     eCancel = 4&
03011     eFinish = 8&
03012     eHelp = 10&
03013
03014     eError = 80000000
03015 End Enum
03016
03017
03018 Private Enum COMMANDLINE_RETURN
03019     eCommandLineError = 0&
03020     eRebootForMSI = 1&
03021     eRebootForConfig = 2&
03022     eManualConfig = 3&
03023     eMigrate2000Data = 4&
03024
03025     eCommandLineNOError = 10&
03026 End Enum
03027
03028 Public Enum MSI_REBOOT
03029     eRebootNotNeeded = 0&
03030     eRebootNeeded = 1&
03031     eMSIInstallFailed = 2&
03032 End Enum
03033
03034
03035 Private m_ofrmTaskbarProxy As frmTaskbarProxy

```

```

03036
03037 Public Property Get TaskbarProxy() As frmTaskbarProxy
03038     Set TaskbarProxy = m_ofrmTaskbarProxy
03039 End Property

```

```

03040
03041 Private Sub Main()
03042
03043     Dim bCancel As Boolean, bCheckIfMSIsNeeded As Boolean, bServerInstalled As Boolean
03044     Dim sServerName$, sUserID$, sPassword$, sJetDBPath$, bNTAuthentication As Boolean, bDropExistingDB As
» Boolean
03045     Dim oPicture As StdPicture, rc As RECT
03046     Dim eCommandLineReturn As COMMANDLINE_RETURN
03047
03048     Dim ofrmWelcome As frmWelcome, ofrmInstallMSI As frmInstallMSI, ofrmGetServerInfo As frmGetServerInfo
03049     Dim ofrmReadyToInstallMSDE As frmReadyToInstallMSDE, ofrmInstallMSDE As frmInstallMSDE, ofrmConfigureMSDE
» As frmConfigureMSDE
03050
03051
03052     Set m_ofrmTaskbarProxy = New frmTaskbarProxy
03053     With m_ofrmTaskbarProxy
03054         .Move -.Width, -150000, 0, 0
03055         .Show
03056     End With
03057
1

```

```

03058 1 ' setup the rect.... Width in twips = 6615 | Height in twips = 4575
03059 With Screen
03060 rc.Left = ((.Width / .TwipsPerPixel X) / 2) - ((6615 / .TwipsPerPixel X) / 2)
03061 rc.Top = ((.Height / .TwipsPerPixel Y) / 2.5) - ((4575 / .TwipsPerPixel Y) / 2)
03062 End With
03063
03064
03065 eCommandlineReturn = ProcessCommandline(sServerName, sUserID, sPassword, bNTAuthentication)
03066 ' valid command lines:
03067 ;
03068 ' /Config <registry key pointer to the following encrypted data> -->
> "ServerName|UserID|Password|bNTAuthentication"
03069 ' this switch is used when a reboot is needed after MSDE has been installed.
03070 ' the configuration info is encrypted and stored in the registry for retrieval upon reboot.
03071 ' the app is added to the "RUN" key in the registry so that it will be restarted upon reboot.
03072 ;
03073 ' /MSI ' no other data for this switch
03074 ' tells the utility that MSI was installed and a reboot was needed
03075 ;
03076 ' /MANUALCONFIG ' no other data for this switch
03077 ' allows an installed MSDE instance to be configured when the reboot configuration
> process fails
03078
03079
03080 If eCommandlineReturn <> eCommandlineError Then
03081
03082 { Select Case eCommandlineReturn
03083 { Case eRebootForConfig
03084 bServerInstalled = True
03085
03086 GoTo Configuration
03087
03088 }
03089 { Case eManualConfig
03090 Set ofrmGetServerInfo = New frmGetServerInfo
03091
03092 { Select Case ofrmGetServerInfo.DisplayDialog(OptionPicture, VarPtr(rc), sServerName, sUserID, sPassword,
> bNTAuthentication)
03093 { Case eNext
03094 bServerInstalled = True
03095 GoTo Configuration
03096
03097 { Case eCancel, eBack: bCancel = True
03098 End Select
03099
03100 Set ofrmGetServerInfo = Nothing
03101
03102 End Select
03103
03104 Back1:
03105 If bCancel = False Then
03106 Set ofrmWelcome = New frmWelcome
03107
03108 { Select Case ofrmWelcome.DisplayDialog(OptionPicture, VarPtr(rc))
03109 { Case eNext
03110 If Len(sServerName) = 0 Then sServerName = DEFAULT_SERVER_INSTANCE_NAME
03111 If Len(sUserID) = 0 Then sUserID = DEFAULT_USER_ID
03112 If Len(sPassword) = 0 Then sPassword = DEFAULT_PASSWORD
03113
03114 bCheckIfMSIIsNeeded = True
03115
03116 }
03117 { Case eCancel: bCancel = True
03118 End Select
03119
03120 Set ofrmWelcome = Nothing
03121 End If
03122
03123
03124 If bCheckIfMSIIsNeeded Then
03125 If IsMSINeeded() Then
03126 Set ofrmInstallMSI = New frmInstallMSI
03127
03128 { Select Case ofrmInstallMSI.DisplayDialog(OptionPicture, VarPtr(rc))
03129 { Case eFinish: bCancel = True
03130 { Case eCancel: bCancel = True
03131 End Select
03132
03133 Set ofrmInstallMSI = Nothing

```

```

03134 1 2 3 4 End If
03135 End If
03136
03137
03138 Back2:
03139 If bCancel = False Then
03140     Set ofrmGetServerInfo = New frmGetServerInfo
03141
03142     Select Case ofrmGetServerInfo.Di spl ayDi al og(oPi ctur e, VarPtr(rc), sServerName, sUserI D, sPassword,
>     bNTAuthenti cati on)
03143     } Case eBack: GoTo Back1
03144     } Case eCancel: bCancel = True
03145     } End Select
03146
03147     Set ofrmGetServerInfo = Nothing
03148 End If
03149
03150
03151 ' section to install and configure MSDE...
03152 If bCancel = False Then
03153     Set ofrmReadyToInstallMSDE = New frmReadyToInstallMSDE
03154
03155     Select Case ofrmReadyToInstallMSDE.Di spl ayDi al og(oPi ctur e, VarPtr(rc))
03156     } Case eBack: GoTo Back2
03157     } Case eCancel: bCancel = True
03158
03159     } Case eError
03160     }     bCancel = True
03161
03162     } End Select
03163
03164     Set ofrmReadyToInstallMSDE = Nothing
03165
03166
03167     If bCancel = False Then
03168         Set ofrmInstallMSDE = New frmInstallMSDE
03169
03170         Select Case ofrmInstallMSDE.Di spl ayDi al og(oPi ctur e, VarPtr(rc), sServerName, sUserI D, sPassword,
>         bNTAuthenti cati on)
03171         } Case eNext: bServerInstalled = True
03172         } Case eBack: GoTo Back2 '--- this option is only enabled when an error occurs that may be corrected
>         }     by changing the server data
03173         } Case eCancel: bCancel = True
03174         } End Select
03175
03176         Set ofrmInstallMSDE = Nothing
03177     End If
03178 End If
03179
03180
03181 Configurati on:
03182 If (bCancel = False) And (bServerInstalled = True) Then
03183     Set ofrmConfigureMSDE = New frmConfigureMSDE
03184
03185     Select Case ofrmConfigureMSDE.Di spl ayDi al og(oPi ctur e, VarPtr(rc), sServerName, sUserI D, sPassword,
>     bNTAuthenti cati on)
03186     } Case eFi ni sh
03187     } Case eBack 'can't go back
03188     } Case eCancel: bCancel = True
03189     } End Select
03190
03191     Set ofrmConfigureMSDE = Nothing
03192 End If
03193
03194 End If 'ProcessCommandLi ne
03195
03196 ExitNow:
03197     Unload m_ofrmTaskbarProxy
03198     Set m_ofrmTaskbarProxy = Nothing
03199     Exit Sub
03200
03201 EH:
03202 With Err
03203     MsgBox "The following error occurred: " & .Descripti on & " Number: " & .CStr(.Number) & " Source: " &
>     .Source, vbExcl amati on, nvMSGBOXCAPTION
03204
03205     .Clear
03206 End With
03207

```

```

03208 1 Resume ExitNow
03209
03210 End Sub

```

```

03211
03212 Private Function ProcessCommandline(ByRef sServerName$, ByRef sUserID$, ByRef sPassword$, ByRef
» bNTAuthentication As Boolean) As COMMANDLINE_RETURN
03213
03214 Dim sCommand$, nSwitchLocation&, sRegKey$, oRegistry As CRegistry, bInvalidCommandline As Boolean,
» bSwitchProcessed As Boolean
03215
03216 sCommand = LCase$(Command$)
03217
03218 If Len(sCommand) Then
03219     nSwitchLocation = InStr(sCommand, "/config")
03220
03221     If nSwitchLocation Then
03222         nSwitchLocation = nSwitchLocation + 7
03223         If Len(sCommand) > nSwitchLocation Then
03224             sRegKey = Trim$(Right$(sCommand, Len(sCommand) - nSwitchLocation))
03225
03226             If Len(sRegKey) Then
03227                 Set oRegistry = New CRegistry
03228
03229                 With oRegistry
03230                     .Key = HKEY_LOCAL_MACHINE
03231
03232                     sCommand = Encrypt(CStr(.ReadOnlyReadSetting("SOFTWARE\NV", sRegKey, vbNullString, REG_BINARY))
» )
03233                 End With
03234
03235                 Set oRegistry = Nothing
03236
03237                 Call RemoveCommandlineKey(sRegKey)
03238
03239                 If ParseServerInfo(sCommand, sServerName, sUserID, sPassword, bNTAuthentication) Then
03240                     bSwitchProcessed = True
03241
03242                     ProcessCommandline = eRebootForConfig
03243                 Else
03244                     ' missing command line data
03245                     bInvalidCommandline = True
03246                 End If
03247             Else
03248                 bInvalidCommandline = True
03249             End If
03250         Else
03251             bInvalidCommandline = True
03252         End If
03253     End If
03254
03255
03256     nSwitchLocation = InStr(sCommand, "/msi")
03257
03258     If nSwitchLocation Then
03259         bSwitchProcessed = True
03260
03261         ProcessCommandline = eRebootForMSI
03262     End If
03263
03264
03265     nSwitchLocation = InStr(sCommand, "/manual config")
03266
03267     If nSwitchLocation Then
03268         bSwitchProcessed = True
03269
03270         ProcessCommandline = eManualConfig
03271     End If
03272
03273
03274
03275     ' more command switches here...
03276
03277
03278     If (bInvalidCommandline = True) Or (bSwitchProcessed = False) Then
03279         MsgBox "Invalid command line parameters." & vbCrLf & vbCrLf &
03280             & "The following are valid parameters:" & vbCrLf & vbCrLf &
03281             & "/Config <configuration data>" & vbCrLf & vbCrLf &
03282             & "/MSI" & vbCrLf & vbCrLf &
03283             & "/MANUALCONFIG"

```

```

03284 1 2 3
03285     ProcessCommandline = eCommandlineError
03286     End If
03287
03288     Else
03289     ProcessCommandline = eCommandlineNOError
03290     End If
03291
03292
03293     Call RemoveRunKey
03294
03295 End Function

```

```

03296
03297 Private Function ParseServerInfo(ByVal sCommandline$, ByRef sServerName$, ByRef sUserID$, ByRef sPassword$,
»   ByRef bNTAuthentication As Boolean) As Boolean
03298
03299     Dim i&, sTemp$, nLastSepPosition&, nItemCount&
03300
03301     If Len(sCommandline) Then
03302
03303         sCommandline = sCommandline & "|" '-- add a trailing pipe to make parsing easier
03304
03305         For i = 1 To Len(sCommandline)
03306             If Mid$(sCommandline, i, 1) = "|" Then
03307                 nLastSepPosition = nLastSepPosition + 1
03308                 sTemp = Mid$(sCommandline, nLastSepPosition, i - nLastSepPosition)
03309
03310                 Select Case nItemCount
03311                 Case 0: sServerName = sTemp
03312                 Case 1: sUserID = sTemp
03313                 Case 2: sPassword = sTemp
03314                 Case 3: bNTAuthentication = CBool(sTemp): ParseServerInfo = True
03315                 End Select
03316
03317                 nLastSepPosition = i
03318                 nItemCount = nItemCount + 1
03319             End If
03320         Next
03321     End If
03322
03323 End Function

```

```

03324
03325 Public Sub ChangePassword(ByVal sServerName$, ByVal sLogi nID$, ByVal sOldPassword$, ByVal sNewPassword$)
03326
03327     Dim oServer As SQLDMO.SQLServer
03328     Set oServer = New SQLDMO.SQLServer
03329
03330     With oServer
03331         .Connect sServerName, sLogi nID, sOldPassword
03332
03333         With .Logi ns(sLogi nID)
03334             .SetPassword sOldPassword, sNewPassword
03335         End With
03336
03337         .Disconnect
03338     End With
03339
03340     Set oServer = Nothing
03341
03342 End Sub

```

```

03343
03344 Public Sub ChangePasswordNTAuth(ByVal sServerName$, ByVal sLogi nID$, ByVal sOldPassword$, ByVal
»   sNewPassword$)
03345
03346     Dim oServer As SQLDMO.SQLServer
03347     Set oServer = New SQLDMO.SQLServer
03348
03349     With oServer
03350         .Logi nSecure = True
03351
03352         .Connect sServerName
03353
03354         With .Logi ns(sLogi nID)
03355             .SetPassword sOldPassword, sNewPassword
03356         End With
03357
03358         .Disconnect
03359     End With

```

```

03360 1
03361     Set oServer = Nothing
03362
03363 End Sub
-----
03364
03365 Public Sub AddSQLServerLogin(ByVal sServerName$, ByVal sAdminLogin$, ByVal sAdminPassword$, ByVal
»   sLoginName$, ByVal sLoginPassword$, _
    Optional ByVal eLoginType As SQLDMO_LOGIN_TYPE = SQLDMOLogin_Standard)
03366
03367     Dim oServer As SQLDMO.SQLServer, oLogin As SQLDMO.Login
03368
03369     'Instantiate SQLServer object
03370     Set oServer = New SQLDMO.SQLServer
03371     With oServer
03372         .Connect sServerName, sAdminLogin, sAdminPassword
03373
03374         Set oLogin = New SQLDMO.Login
03375         With oLogin
03376             .Name = sLoginName
03377             .Type = eLoginType
03378             Call .SetPassword(vbNullString, sLoginPassword)
03379         End With
03380
03381         .Logins.Add oLogin
03382
03383         Set oLogin = Nothing
03384
03385         .Disconnect
03386     End With
03387
03388     Set oServer = Nothing
03389
03390 End Sub
-----
03392
03393 Public Sub AddUserToDatabase(ByVal sServerName$, ByVal sAdminLogin$, ByVal sAdminPassword$, ByVal
»   sLoginName$, ByVal sLoginPassword$, ByVal sDBName$)
03394
03395     Dim oServer As SQLDMO.SQLServer, oUser As SQLDMO.User
03396
03397     'Instantiate SQLServer object
03398     Set oServer = New SQLDMO.SQLServer
03399     With oServer
03400         'If bNTAuthentication Then .LoginSecure = True
03401
03402         .Connect sServerName, sAdminLogin, sAdminPassword
03403
03404         'Instantiate the new User and assign its Name and Login properties.
03405         Set oUser = New SQLDMO.User
03406
03407         oUser.Login = sLoginName
03408
03409         .Databases(sDBName).Users.Add oUser
03410
03411         Set oUser = Nothing
03412
03413         .Disconnect
03414     End With
03415
03416     Set oServer = Nothing
03417
03418 End Sub
-----
03419
03420 Public Sub AddLoginToServerRole(ByVal sServerName$, ByVal sAdminLogin$, ByVal sAdminPassword$, ByVal
»   sLoginName$, ByVal sRoleName$)
03421
03422     Dim oServer As SQLDMO.SQLServer
03423
03424     'Instantiate SQLServer object
03425     Set oServer = New SQLDMO.SQLServer
03426     With oServer
03427         'If bNTAuthentication Then .LoginSecure = True
03428
03429         .Connect sServerName, sAdminLogin, sAdminPassword
03430
03431         Call .ServerRoles(sRoleName).AddMember(sLoginName)
03432
03433         .Disconnect
03434 1 End With

```

```

1
03435
03436     Set oServer = Nothing
03437
03438 End Sub
-----
03439
03440 Public Function GetSQLServerVersion(ByVal sServerName$, ByVal sAdminLogin$, ByVal sAdminPassword$, ByVal
» bNTAuthentication As Boolean) As Long
03441     Dim oServer As SQLDMO.SQLServer
03442
03443     ' Instantiate SQLServer object
03444     Set oServer = New SQLDMO.SQLServer
03445     With oServer
03446         If bNTAuthentication Then .LoginSecure = True
03447
03448         .Connect sServerName, sAdminLogin, sAdminPassword
03449
03450         GetSQLServerVersion = .VersionMajor
03451
03452         .Disconnect
03453     End With
03454     Set oServer = Nothing
03455
03456 End Function
-----
03459
03460 Private Function IsSAPasswordOK(ByVal sServerName$, ByVal sUserID$, ByVal sPassword$) As Boolean
03461
03462     On Error GoTo EH
03463
03464     Dim oServer As SQLDMO.SQLServer, oLogin As SQLDMO.Login
03465     Set oServer = New SQLDMO.SQLServer
03466
03467     With oServer
03468         ' Debug.Print .VersionMajor
03469         ' Debug.Print .VersionMinor
03470         ' Debug.Print .VersionString
03471
03472         .Connect sServerName, sUserID, sPassword
03473
03474         ' For Each oLogin In .Logins
03475         '     cbSQLServerLogins.AddItem oLogin.Name
03476         ' Next
03477
03478         .Disconnect
03479     End With
03480
03481     Set oServer = Nothing
03482
03483     IsSAPasswordOK = True
03484
03485 ExitNow:
03486     Exit Function
03487
03488 EH:
03489     Resume ExitNow
03490 End Function
-----
03492
03493 Public Function DoesSQLServerExist(ByVal sServerName$) As Boolean
03494
03495     On Error GoTo EH
03496
03497     Dim oApp As SQLDMO.Application
03498
03499     Set oApp = New SQLDMO.Application
03500
03501     DoesSQLServerExist = (oApp.ListAvailableSQLServers().FindName(sServerName) <> 0)
03502
03503 ExitNow:
03504     Set oApp = Nothing
03505     Exit Function
03506
03507 EH:
03508     Resume ExitNow
03509 End Function

```

```

03511
03512 Public Function GetSQLServerRolIName(ByVal eRolIType As SQL_SERVER_ROLLS) As String
03513
03514     Select Case eRolIType
03515     } Case eSystem_Administrators: GetSQLServerRolIName = "sysadmin"
03516     } Case eSecurity_Administrators: GetSQLServerRolIName = "securityadmin"
03517     } Case eServer_Administrators: GetSQLServerRolIName = "serveradmin"
03518     } Case eSetup_Administrators: GetSQLServerRolIName = "setupadmin"
03519     } Case eProcess_Administrators: GetSQLServerRolIName = "processadmin"
03520     } Case eDisk_Administrators: GetSQLServerRolIName = "diskadmin"
03521     } Case eDatabase_Creators: GetSQLServerRolIName = "dbcreator"
03522     } Case eBulk_Insert_Administrators: GetSQLServerRolIName = "bulkadmin"
03523     End Select
03524
03525 End Function

```

```

03526
03527 Private Function IsMSINeeded() As Boolean
03528
03529     ' version shipped with MSDE install --> 1.10.1029.1
03530
03531     If IsCallExported("msi", "DllGetVersion") Then
03532         Dim pDVI As DLLVERSIONINFO
03533
03534         pDVI.cbSize = Len(pDVI)
03535
03536         If DllGetVersion(pDVI) = NOERROR Then
03537             With pDVI
03538             } If .dwMajorVersion < 1 Then
03539                 IsMSINeeded = True
03540             } ElseIf (.dwMajorVersion = 1) And (.dwMinorVersion < 10) Then
03541                 IsMSINeeded = True
03542             } ElseIf .dwBuildNumber < 1029 Then
03543                 IsMSINeeded = True
03544             } End If
03545             End With
03546         } Else
03547             IsMSINeeded = True
03548         } End If
03549     } Else
03550         IsMSINeeded = True
03551     } End If
03552
03553 End Function

```

```

03554
03555 Public Function DBNameIsOK(ByVal sDBName$) As Boolean
03556
03557     Dim i &
03558
03559     If Len(sDBName) Then
03560         ' default is true...
03561         DBNameIsOK = True
03562
03563         For i = 1 To Len(sDBName)
03564             Select Case Asc(Mid$(sDBName, i, 1))
03565             } Case 48 To 57, 65 To 90, 95, 97 To 122
03566                 ' do nothing, these chars are OK
03567             } Case Else
03568                 DBNameIsOK = False
03569                 Exit For
03570             } End Select
03571         Next
03572     } End If
03573
03574 End Function

```

```

03576
03577 Public Sub SetRunKey(ByVal sCommandSwitch$, ByVal sRegKey$)
03578
03579     Dim sPath$, oRegistry As CRegistry
03580     Set oRegistry = New CRegistry
03581
03582     sPath = AddBackslashToPath(App.Path) & "MSDE_Installer.exe"
03583
03584     sPath = ReturnShortPathName(sPath)
03585
03586
03587     If Len(sCommandSwitch) Then sCommandSwitch = " " & sCommandSwitch
03588     If Len(sRegKey) Then sRegKey = " " & sRegKey

```

```

1
03589
03590
03591   With oRegistry
03592     . Key = HKEY_LOCAL_MACHINE
03593
03594     . WriteSetting "SOFTWARE\Microsoft\Windows\CurrentVersion\Run", "MSDE Instance Installer", sPath &
»     sCommandSwitch & sRegKey
03595   End With
03596
03597   Set oRegistry = Nothing
03598
03599 End Sub
-----
03600
03601 Public Sub SetCommandLineKey(ByVal sRegKey$, ByVal sCommandLine$)
03602
03603   Dim oRegistry As CRegistry
03604   Set oRegistry = New CRegistry
03605
03606   With oRegistry
03607     . Key = HKEY_LOCAL_MACHINE
03608
03609     . WriteSetting "SOFTWARE\" & COMPANY_NAME, sRegKey, Encrypt(sCommandLine), REG_BINARY
03610   End With
03611
03612   Set oRegistry = Nothing
03613
03614 End Sub
-----
03615
03616 Public Sub RemoveRunKey()
03617
03618   ' set RunOnce registry entry and call exit windows
03619   Dim oRegistry As CRegistry
03620   Set oRegistry = New CRegistry
03621
03622   With oRegistry
03623     . Key = HKEY_LOCAL_MACHINE
03624
03625     . DeleteSetting "SOFTWARE\Microsoft\Windows\CurrentVersion\Run", "MSDE Instance Installer"
03626   End With
03627
03628   Set oRegistry = Nothing
03629
03630 End Sub
-----
03631
03632 Public Sub RemoveCommandLineKey(ByVal sRegKey$)
03633
03634   ' set RunOnce registry entry and call exit windows
03635   Dim oRegistry As CRegistry
03636   Set oRegistry = New CRegistry
03637
03638   With oRegistry
03639     . Key = HKEY_LOCAL_MACHINE
03640
03641     . DeleteSetting "SOFTWARE\" & COMPANY_NAME, sRegKey
03642   End With
03643
03644   Set oRegistry = Nothing
03645
03646 End Sub
-----
03647
03648 Public Function ConfigureSQLServer(ByVal sServerName$, ByVal sUserID$, ByVal sPassword$, ByVal
» bNTAuthentication As Boolean, Optional oCallback As Form) As Boolean
03649
03650   Dim i&, sMgrPath$, sSvrPath$, sComputerName$, sComspecPath$, sCmdLine$, sStartDir$, blsDefaultInstance As
» Boolean, bServiceStarted As Boolean
03651   Dim nExitCode&, nWaitState&, udtStartInf As STARTUPINFO, udtProcInf As PROCESS_INFORMATION
03652   Dim hInst&, hSQLManager&, hCheckbox&, sShortPath$, bRegisterFailed As Boolean
03653
03654   Const BINN_PATH As String = "C:\Program Files\Microsoft SQL Server\80\Tool\Bin\bin"
03655
03656   Screen.MousePointer = vbHourglass
03657
03658
03659   On Error GoTo EH
03660
03661   hSQLManager = FindWindowStr("SQLManagerApplicationClass", "SQL Server Service Manager")
03662
1

```

```

1
03663
03664 If hSQLManager = API_FALSE Then
03665     sMgrPath = ReturnShortPathName(BINN_PATH & "sqlmangr.exe")
03666
03667     If FileExists(sMgrPath) Then
03668         With udtStartInf
03669             .cb = Len(udtStartInf)
03670             .lpTitle = vbNullString
03671         End With
03672
03673     If CreateProcess(sMgrPath, "/n", 0&, 0&, API_FALSE, NORMAL_PRIORITY_CLASS, ByVal 0&, vbNullString,
»     udtStartInf, udtProcInf) Then
03674
03675         Do
03676             Yield
03677             nWaitState = WaitForInputIdle(udtProcInf.hProcess, 500&)
03678         Loop Until (nWaitState = WAIT_OBJECT_0) Or (nWaitState = WAIT_FAILED)
03679
03680         Call CloseHandle(udtProcInf.hThread)
03681         Call CloseHandle(udtProcInf.hProcess)
03682
03683         If Not oCallbacDisplay Is Nothing Then
03684             With oCallbacDisplay
03685                 .ConfigurationCallback "SQL Manager started successfully..."
03686                 .Refresh
03687             End With
03688         End If
03689     End If
03690
03691     hSQLManager = FindWindowStr("SQLManagerApplicationClass", "SQL Server Service Manager")
03692
03693     If hSQLManager Then GoTo SetAutoStart
03694
03695 End If
03696
03697 Else
03698 SetAutoStart:
03699
03700     hCheckbox = FindWindowEx(hSQLManager, 0&, "Button", "&Auto-start service when OS starts")
03701
03702     If hCheckbox Then
03703         If SendMessage(hCheckbox, BM_GETCHECK, ByVal 0&, ByVal 0&) <> BST_CHECKED Then
03704             Call SendMessage(hCheckbox, BM_SETCHECK, ByVal BST_CHECKED, ByVal 0&)
03705         End If
03706     End If
03707
03708     Call Sleep(1000&)
03709
03710     Call PostMessage(hSQLManager, WM_CLOSE, ByVal 0&, ByVal 0&)
03711 End If
03712
03713
03714 If UCase$(sServerName) = "DEFAULT" Then blsDefaultInstance = True
03715
03716
03717 sComputerName = ReturnComputerName()
03718
03719
03720
03721 If GetOSVersion() < [eNT4.0] Then
03722     sSvrPath = ReturnShortPathName(BINN_PATH & "scm.exe")
03723
03724     If FileExists(sSvrPath) Then
03725         sSvrPath = sSvrPath & " -Action 1 -Silent 1 -Server " & sComputerName & " -Service MSSQL"
03726
03727         If blsDefaultInstance = False Then sSvrPath = sSvrPath & "$" & sServerName
03728
03729         With udtStartInf
03730             .cb = Len(udtStartInf)
03731             .lpTitle = vbNullChar
03732         End With
03733
03734     If CreateProcess(vbNullString, sSvrPath, 0&, 0&, API_FALSE, NORMAL_PRIORITY_CLASS, ByVal 0&,
»     vbNullString, udtStartInf, udtProcInf) Then
03735
03736         Do
03737             Yield
03738             nWaitState = WaitForInputIdle(udtProcInf.hProcess, 500&)
03739         Loop Until (nWaitState = WAIT_OBJECT_0) Or (nWaitState = WAIT_FAILED)

```

```

1 2 3 4
03740 Call CloseHandle(udtProcInf.hThread)
03741 Call CloseHandle(udtProcInf.hProcess)
03742
03743     bServiceStarted = True
03744 End If
03745 End If
03746
03747 Else
03748     sComspecPath = Environ$("COMSPEC")
03749
03750     If FileExists(sComspecPath) Then
03751         sCmdLine = sComspecPath & " /C net start "
03752
03753         Call ReturnFileName(sComspecPath, sStartDir)
03754
03755         Select Case blsDefaultInstance
03756         Case True: sCmdLine = sCmdLine & "MSSQLSERVER"
03757         Case False: sCmdLine = sCmdLine & "MSSQL$" & sServerName
03758         End Select
03759
03760
03761         With udtStartInf
03762             .cb = Len(udtStartInf)
03763             .wShowWindow = SW_HIDE
03764             .lpTitle = vbNullString
03765         End With
03766
03767         If CreateProcess(vbNullString, sCmdLine, 0&, 0&, API_FALSE, NORMAL_PRIORITY_CLASS, ByVal 0&,
03768             sStartDir, udtStartInf, udtProcInf) Then
03769             Do
03770                 Yield
03771                 nWaitState = WaitForSingleObject(udtProcInf.hProcess, 500&)
03772             Loop Until (nWaitState = WAIT_OBJECT_0) Or (nWaitState = WAIT_FAILED)
03773
03774             ' Call GetExitCodeProcess(udtProcInf.hProcess, nExitCode)
03775
03776             Call CloseHandle(udtProcInf.hThread)
03777             Call CloseHandle(udtProcInf.hProcess)
03778
03779             bServiceStarted = True
03780         End If
03781     End If
03782 End If
03783
03784
03785 If bServiceStarted Then
03786     If Not oCallbacDisplay Is Nothing Then
03787         With oCallbacDisplay
03788             .ConfigurationCallback "SQL Server service started successfully..."
03789             .Refresh
03790         End With
03791     End If
03792
03793
03794
03795     If (blsDefaultInstance = False) And (InStr(sServerName, "\") = 0) Then
03796         sComputerName = sComputerName & "\" & sServerName
03797     ElseIf (blsDefaultInstance = False) Then
03798         sComputerName = sServerName
03799     End If
03800
03801
03802     If bNTAuthentication = False Then
03803         If Not oCallbacDisplay Is Nothing Then oCallbacDisplay.ConfigurationCallback "Setting sa
03804             password..."
03805
03806         Call ChangePassword(sComputerName, "sa", vbNullString, SA_PASSWORD)
03807
03808         If Not oCallbacDisplay Is Nothing Then oCallbacDisplay.ConfigurationCallback "Setting up user
03809             login..."
03810
03811         Call AddSQLServerLogin(sComputerName, "sa", SA_PASSWORD, sUserID, sPassword)
03812
03813         If Not oCallbacDisplay Is Nothing Then oCallbacDisplay.ConfigurationCallback "Setting user
03814             password..."
03815
03816         Call AddLoginToServerRole(sComputerName, "sa", SA_PASSWORD, sUserID, GetSQLServerRoleName(
03817             eSystemAdministrators))
03818     End If
03819 End If
1 2 3

```

```

03814 1 2 3
03815 } Else
03816   If Not oCall bacDi spl ay Is Nothing Then oCall bacDi spl ay. Configurati onCall back "Setting sa
   password..."
»
03817   Call ChangePasswordNTAuth(sComputerName, "sa", vbNullString, SA_PASSWORD)
03818   End If
03819
03820
03821   ConfigureSQLServer = True
03822   End If
03823
03824   Screen.MousePointer = vbDefault
03825
03826
03827 Exit Now:
03828   Exit Function
03829
03830 EH:
03831   Screen.MousePointer = vbDefault
03832   With Err
03833     MsgBox "The following error ocured during database configuration: " & .Description & " " & CStr(.
     Number), vbExclamation, nvMSGBOXCAPTION
»
03834     .Clear
03835   End With
03836
03837   Resume ExitNow
03838
03839 End Function
03840


---


03841 Public Function Encrypt(ByVal sSecret$) As String
03842   ' simple encryption
03843
03844   Dim i&, char&
03845
03846   If Right$(sSecret, 1) = Chr$(255) Then ' since Trim$ is used on encrypted strings after they come
03847     Mid$(sSecret, Len(sSecret), 1) = " " ' out of this function ending spaces can be clipped off. So
03848     End If ' before the function exits it changes the last char to 255 if
03849     ' it happens to be a space (chr 32).
03850     char = 1
03851
03852   For i = 1 To Len(sSecret) ' xor the chars in sSecret against those in sMatrix
03853     Mid$(sSecret, i, 1) = Chr$(Asc(Mid$(sSecret, i, 1)) Xor char)
03854
03855     char = char + 3
03856     If char > 254 Then char = char - 254
03857   Next
03858
03859   If Right$(sSecret, 1) = " " Then ' if the last char is a space change it to chr 255.
03860     Mid$(sSecret, Len(sSecret), 1) = Chr$(255)
03861   End If
03862
03863   Encrypt = sSecret ' return the string
03864
03865 End Function
03866


---



```

```

03868 Option Explicit
03869 ' demo project showing how to create a custom MSDE install
03870 ' by Bryan Stafford of New Vision Software - newvision@mvps.org
03871 ' this demo is released into the public domain "as is" without
03872 ' warranty or guaranty of any kind. In other words, use at
03873 ' your own risk.
03874 '
03875 ' Copyright © 2002 Bryan Stafford/New Vision Software
03876 '
03877 ' the code in this demo is copyrighted and may only be used in
03878 ' accordance with the rules set forth in the "House Rules"
03879 ' section of the web page found at http://www.mvps.org/vbvision/
03880 '
03881 ' see the general declararions section of MMain.bas for more info
03882 ' on this project
03883
03884
03885 ' Reg Data Types...
03886 Public Enum EREGTYPE
03887     REG_NONE = 0& ' No value type
03888     REG_SZ = 1& ' Unicode nul terminated string
03889     REG_EXPAND_SZ = 2& ' Unicode nul terminated string
03890     REG_BINARY = 3& ' Free form binary
03891     REG_DWORD = 4& ' 32-bit number
03892     REG_DWORD_LITTLE_ENDIAN = 4& ' 32-bit number (same as REG_DWORD)
03893     REG_DWORD_BIG_ENDIAN = 5& ' 32-bit number
03894     REG_LINK = 6& ' Symbolic Link (unicode)
03895     REG_MULTI_SZ = 7& ' Multiple Unicode strings
03896     REG_RESOURCE_LIST = 8& ' Resource list in the resource map
03897     REG_FULL_RESOURCE_DESCRIPTOR = 9& ' Resource list in the hardware description
03898     REG_RESOURCE_REQUIREMENTS_LIST = 10&
03899 End Enum
03900
03901 Private Const STANDARD_RIGHTS_READ As Long = &H20000
03902 Private Const STANDARD_RIGHTS_WRITE As Long = &H20000
03903 Private Const STANDARD_RIGHTS_EXECUTE As Long = &H20000
03904 Private Const STANDARD_RIGHTS_REQUIRED As Long = &HF0000
03905 Private Const STANDARD_RIGHTS_ALL As Long = &H1F0000
03906
03907 Private Const SYNCHRONIZE As Long = &H100000
03908
03909 ' Reg Create Type Values...
03910 Private Const REG_OPTION_RESERVED As Long = 0& ' Parameter is reserved
03911 Private Const REG_OPTION_NON_VOLATILE As Long = 0& ' Key is preserved when system is rebooted
03912 Private Const REG_OPTION_VOLATILE As Long = 1& ' Key is not preserved when system is rebooted
03913 Private Const REG_OPTION_CREATE_LINK As Long = 2& ' Created key is a symbolic link
03914 Private Const REG_OPTION_BACKUP_RESTORE As Long = 4& ' open for backup or restore
03915
03916 Private Const REG_CREATED_NEW_KEY As Long = &H1& ' New Registry Key created
03917 Private Const REG_OPENED_EXISTING_KEY As Long = &H2& ' Existing Key opened
03918 Private Const REG_WHOLE_HIVE_VOLATILE As Long = &H1& ' Restore whole hive volatile
03919 Private Const REG_REFRESH_HIVE As Long = &H2& ' Unwind changes to last flush
03920 Private Const REG_NOTIFY_CHANGE_NAME As Long = &H1& ' Create or delete (child)
03921 Private Const REG_NOTIFY_CHANGE_ATTRIBUTES As Long = &H2&
03922 Private Const REG_NOTIFY_CHANGE_LAST_SET As Long = &H4& ' Time stamp
03923 Private Const REG_NOTIFY_CHANGE_SECURITY As Long = &H8&
03924 Private Const REG_LEGAL_CHANGE_FILTER As Long = (REG_NOTIFY_CHANGE_NAME Or REG_NOTIFY_CHANGE_ATTRIBUTES Or
    » REG_NOTIFY_CHANGE_LAST_SET Or REG_NOTIFY_CHANGE_SECURITY)
03925 Private Const REG_LEGAL_OPTION As Long = (REG_OPTION_RESERVED Or REG_OPTION_NON_VOLATILE Or
    » REG_OPTION_VOLATILE Or REG_OPTION_CREATE_LINK Or REG_OPTION_BACKUP_RESTORE)
03926
03927
03928 ' Reg Key Security Options
03929 Private Const KEY_QUERY_VALUE As Long = &H1&
03930 Private Const KEY_SET_VALUE As Long = &H2&
03931 Private Const KEY_CREATE_SUB_KEY As Long = &H4&
03932 Private Const KEY_ENUMERATE_SUB_KEYS As Long = &H8&
03933 Private Const KEY_NOTIFY As Long = &H10&
03934 Private Const KEY_CREATE_LINK As Long = &H20&
03935 Private Const KEY_READ As Long = ((STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or KEY_ENUMERATE_SUB_KEYS Or
    » KEY_NOTIFY) And (Not SYNCHRONIZE))
03936 Private Const KEY_WRITE As Long = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY) And (Not
    » SYNCHRONIZE))
03937 Private Const KEY_EXECUTE As Long = (KEY_READ)
03938 Private Const KEY_ALL_ACCESS As Long = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE Or KEY_SET_VALUE Or
    » KEY_CREATE_SUB_KEY Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY Or KEY_CREATE_LINK) And (Not SYNCHRONIZE))
03939
03940
03941 Private Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" (ByVal hKey&, ByVal

```

```

» IpSubKey$) As Long
03942 Private Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValue" (ByVal hKey&, ByVal
» IpValueName$) As Long
03943 Private Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyEx" (ByVal hKey&, ByVal
» IpSubKey$, ByVal ulOptions&, ByVal samDesired&, phkResult&) As Long
03944 Private Declare Function RegQueryValueExStr Lib "advapi32.dll" Alias "RegQueryValueExA" (ByVal hKey&, ByVal
» IpValueName$, ByVal IpReserved&, IpType&, ByVal IpData$, IpcbData&) As Long
03945 Private Declare Function RegQueryValueExInt Lib "advapi32.dll" Alias "RegQueryValueExA" (ByVal hKey&, ByVal
» IpValueName$, ByVal IpReserved&, IpType&, ByVal IpData&, IpcbData&) As Long
03946 Private Declare Function RegQueryValueExByte Lib "advapi32.dll" Alias "RegQueryValueExA" (ByVal hKey&,
» ByVal IpValueName$, ByVal IpReserved&, IpType&, IpData As Byte, IpcbData&) As Long
03947 Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey&)
03948
03949 Private Declare Function RegSetValueExStr Lib "advapi32.dll" Alias "RegSetValueExA" (ByVal hKey&, ByVal
» IpValueName$, ByVal Reserved&, ByVal dwType&, ByVal IpData$, ByVal cbData&)
03950 Private Declare Function RegSetValueExInt Lib "advapi32.dll" Alias "RegSetValueExA" (ByVal hKey&, ByVal
» IpValueName$, ByVal Reserved&, ByVal dwType&, ByVal IpData&, ByVal cbData&)
03951 Private Declare Function RegSetValueExByte Lib "advapi32.dll" Alias "RegSetValueExA" (ByVal hKey&, ByVal
» IpValueName$, ByVal Reserved&, ByVal dwType&, ByVal IpData As Byte, ByVal cbData&)
03952
03953 Private Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias "RegCreateKeyExA" (ByVal hKey&, ByVal
» IpSubKey$, ByVal Reserved&, ByVal IpClass$, ByVal dwOptions&, ByVal samDesired&, ByVal
» IpSecurityAttributes&, phkResult&, IpDwDisposition&) As Long
03954
03955
03956 Private Const ERROR_MORE_DATA As Long = 234&
03957
03958
03959 ' Usage:
03960 ' Dim oRegistryAccess As CRegistry
03961 ' With oRegistryAccess
03962 '     .Key = HKEY_LOCAL_MACHINE
03963 '     sText = .ReadSetting("SOFTWARE\Starwave\Application", "SomeText", "defaultValue")
03964 '     iNum = .ReadSetting("SOFTWARE\Starwave\Application", "SomeNumber", 0, REG_DWORD)
03965 ' End With
03966 ' Set registry = Nothing
03967
03968 Public Enum REGISTRY_KEYS
03969     HKEY_CLASSES_ROOT = &H80000000
03970     HKEY_CURRENT_USER = &H80000001
03971     HKEY_LOCAL_MACHINE = &H80000002
03972     HKEY_USERS = &H80000003
03973 End Enum
03974
03975 Private m_IKey As REGISTRY_KEYS
03976 Private m_IMaxStringSize As Long
03977 Private m_IValueType As Long

```

```

03978
03979 Private Sub Class_Initialize()
03980     m_IKey = HKEY_CURRENT_USER
03981     m_IValueType = REG_SZ
03982     m_IMaxStringSize = 512&
03983 End Sub

```

```

03984
03985 Public Property Let Key(ByVal NewKey As REGISTRY_KEYS)
03986     If (NewKey >= HKEY_CLASSES_ROOT And NewKey <= HKEY_USERS) Then
03987         m_IKey = NewKey
03988     End If
03989 End Property

```

```

03990
03991 Public Property Get Key() As REGISTRY_KEYS
03992     Key = m_IKey
03993 End Property

```

```

03994
03995 Public Property Let DefaultValueType(ByVal NewValueType As Long)
03996     If (NewValueType = REG_SZ Or NewValueType <= REG_EXPAND_SZ Or NewValueType = REG_DWORD) Then
03997         m_IValueType = NewValueType
03998     End If
03999 End Property

```

```

04000
04001 Public Property Get DefaultValueType() As Long
04002     DefaultValueType = m_IValueType
04003 End Property

```

```

04004
04005 Public Property Let MaxStringSize(ByVal Max As Long)
04006     m_IMaxStringSize = Max

```

```

04007 1 End Property
04008
04009 Public Property Get MaxStringSize() As Long
04010     MaxStringSize = m_MaxStringSize
04011 End Property
04012 '
04013 'Public Function ReadOnlyReadSetting(ByVal SubKey As String, ByVal ValueName As String, Optional ByVal
»     Default As Variant, _
04014 '                                     Optional ByVal ValueType As REGTYPE
»     = REG_SZ) As Variant
04015 '     Dim IType As Long, hKey As Long
04016 '     Dim sValue As String, lValue As Long, lSize As Long
04017 '
04018 '     IType = ValueType
04019 '     If (IType <> REG_SZ And IType <> REG_EXPAND_SZ And IType <> REG_DWORD) Then IType = m_lValueType
04020 '
04021 '     If (RegOpenKeyEx(m_lKey, SubKey, 0, KEY_QUERY_VALUE, hKey) = ERROR_SUCCESS) Then
04022 '
04023 '         Select Case IType
04024 '             Case REG_SZ, REG_EXPAND_SZ
04025 '                 sValue = String$(m_MaxStringSize + 1, vbNullChar)
04026 '                 lSize = m_MaxStringSize
04027 '                 If (RegQueryValueExStr(hKey, ValueName, 0, IType, sValue, lSize) = ERROR_SUCCESS) Then
04028 '                     If lSize > 0 Then
04029 '                         lSize = Min(InStr(sValue, vbNullChar), (lSize + 1))
04030 '                         ReadOnlyReadSetting = CVar(Left$(sValue, lSize - 1))
04031 '                     Else
04032 '                         ReadOnlyReadSetting = CVar(vbNullString)
04033 '                     End If
04034 '                 Else
04035 '                     If Not IsMissing(Default) Then
04036 '                         ReadOnlyReadSetting = Default
04037 '                     End If
04038 '                 End If
04039 '
04040 '             Case REG_DWORD
04041 '                 lSize = 4&
04042 '                 If (RegQueryValueExInt(hKey, ValueName, 0, IType, lValue, lSize) = ERROR_SUCCESS) Then
04043 '                     ReadOnlyReadSetting = CVar(lValue)
04044 '                 Else
04045 '                     If Not IsMissing(Default) Then
04046 '                         ReadOnlyReadSetting = Default
04047 '                     End If
04048 '                 End If
04049 '             End Select
04050 '             Call RegCloseKey(hKey)
04051 '
04052 '         Else
04053 '             If Not IsMissing(Default) Then
04054 '                 ReadOnlyReadSetting = Default
04055 '             End If
04056 '         End If
04057 '     End Function
04058
04059 Public Function ReadOnlyReadSetting(ByVal sSubKey$, ByVal sValueName$, Optional ByVal vDefault As Variant,
»     Optional ByVal eValueType As REGTYPE = REG_SZ) As Variant
04060     ReadOnlyReadSetting = InternalReadSetting(sSubKey, sValueName, True, vDefault, eValueType)
04061 End Function
04062
04063 Public Function ReadSetting(ByVal sSubKey$, ByVal sValueName$, Optional ByVal vDefault As Variant, Optional
»     ByVal eValueType As REGTYPE = REG_SZ) As Variant
04064     ReadSetting = InternalReadSetting(sSubKey, sValueName, False, vDefault, eValueType)
04065 End Function
04066
04067 Private Function InternalReadSetting(ByVal sSubKey$, ByVal sValueName$, ByVal bReadOnly As Boolean,
»     Optional ByVal vDefault As Variant, Optional ByVal eValueType As REGTYPE = REG_SZ) As Variant
04068     Dim eType As REGTYPE, hKey&, nAccessType&
04069     Dim sValue$, nValue As Long, nSize&, aByteData() As Byte
04070
04071     If bReadOnly Then
04072         nAccessType = KEY_QUERY_VALUE
04073     Else
04074         nAccessType = KEY_ALL_ACCESS
04075     End If
04076
04077     eType = eValueType
04078 1

```

```

04079 1 If (eType <> REG_SZ) And (eType <> REG_EXPAND_SZ) And (eType <> REG_DWORD) And (eType <> REG_BINARY) Then
»     eType = m_InvalidType
04080
04081 If (RegOpenKeyEx(m_Key, sSubKey, 0, nAccessType, hKey) = ERROR_SUCCESS) Then
04082
04083     Select Case eType
04084     Case REG_SZ, REG_EXPAND_SZ
04085         sValue = String$(m_MaxStringSize + 1, 0)
04086         nSize = m_MaxStringSize
04087         If (RegQueryValueExStr(hKey, sValueName, 0, eType, sValue, nSize) = ERROR_SUCCESS) Then
04088             If nSize > 0 Then
04089                 nSize = Min(InStr(sValue, vbNullChar), (nSize + 1))
04090                 InternalReadSetting = CVar(Left$(sValue, nSize - 1))
04091             Else
04092                 InternalReadSetting = CVar(vbNullString)
04093             End If
04094         Else
04095             If Not IsMissing(vDefault) Then InternalReadSetting = vDefault
04096         End If
04097
04098     Case REG_DWORD
04099         nSize = 4&
04100         If (RegQueryValueExInt(hKey, sValueName, 0, eType, nValue, nSize) = ERROR_SUCCESS) Then
04101             InternalReadSetting = CVar(nValue)
04102         Else
04103             If Not IsMissing(vDefault) Then InternalReadSetting = vDefault
04104         End If
04105
04106     Case REG_BINARY
04107         nSize = 1000
04108         ReDim aByteData(nSize - 1) As Byte
04109
04110     Do
04111         Select Case RegQueryValueExByte(hKey, sValueName, 0, eType, aByteData(0), nSize)
04112         Case ERROR_SUCCESS
04113             If UBound(aByteData) > (nSize - 1) Then ReDim Preserve aByteData(nSize - 1) As Byte
04114
04115             InternalReadSetting = aByteData
04116         Exit Do
04117         Case ERROR_MORE_DATA
04118             ReDim aByteData(nSize - 1) As Byte
04119         Case Else
04120             If Not IsMissing(vDefault) Then InternalReadSetting = vDefault
04121             Exit Do
04122         End Select
04123     Loop
04124
04125 End Select
04126
04127 Call RegCloseKey(hKey)
04128
04129 Else
04130     If Not IsMissing(vDefault) Then InternalReadSetting = vDefault
04131 End If
04132
04133 End Function

```

```

04142
04143 Public Function WriteSetting(ByVal sSubKey$, ByVal sValueName$, ByVal vValueData As Variant, Optional ByVal
»     eValueType As EREGTYPE = REG_SZ) As Boolean
04144
04145     Dim eType As EREGTYPE, hKey As Long
04146     Dim sValue As String, nValue As Long, aByteData() As Byte, bKeyOpened As Boolean
04147
04148     eType = eValueType
04149     If (eType <> REG_SZ) And (eType <> REG_EXPAND_SZ) And (eType <> REG_DWORD) And (eType <> REG_BINARY) Then
»         eType = m_InvalidType
04150
04151     If (RegOpenKeyEx(m_Key, sSubKey, 0, KEY_ALL_ACCESS, hKey) <> ERROR_SUCCESS) Then hKey = RegCreateKey(
»         sSubKey)
04152
04153 1 If hKey Then bKeyOpened = True

```

```

04154 1
04155     If bKeyOpened Then
04156     Select Case eType
04157     Case REG_SZ, REG_EXPAND_SZ
04158         sValue = CStr(vValueData)
04159         If (RegSetValueExStr(hKey, sValueName, 0, eType, sValue, CLng(Len(sValue))) = ERROR_SUCCESS) Then
04160             WriteSetting = True
04161         End If
04162     Case REG_DWORD
04163         nValue = CLng(vValueData)
04164         If (RegSetValueExInt(hKey, sValueName, 0, REG_DWORD, nValue, CLng(Len(nValue))) = ERROR_SUCCESS) Then
04165             WriteSetting = True
04166         End If
04167     Case REG_BINARY
04168         ' ReDim aByteData(LenB(vValueData) + 1) As Byte
04169         Select Case VarType(vValueData)
04170         Case vbString: vValueData = vValueData & StrConv(vbNullChar, vbFromUnicode)
04171         End Select
04172         aByteData = vValueData
04173         If (RegSetValueExByte(hKey, sValueName, 0, REG_BINARY, aByteData(0), UBound(aByteData)) =
04174             ERROR_SUCCESS) Then
04175             WriteSetting = True
04176         End If
04177     End Select
04178     Call RegCloseKey(hKey)
04179 End If
04180
04181
04182
04183
04184
04185
04186
04187
04188
04189 End Function
04190
04191 Public Function DeleteSetting(ByVal sKey$, ByVal sValue$) As Boolean
04192
04193     Dim hKey&
04194
04195     ' First, open up the registry key which holds the value to delete.
04196     If RegOpenKeyEx(m_IKey, sKey, 0, KEY_ALL_ACCESS, hKey) = ERROR_SUCCESS Then
04197         ' Now delete the desired value from the key.
04198         DeleteSetting = (RegDeleteValue(hKey, sValue) = ERROR_SUCCESS)
04199     End If
04200     Call RegCloseKey(hKey)
04201 End If
04202
04203 End Function
04204
04205 Private Function RegCreateKey(ByVal SubKey As String) As Long
04206     Dim hNewKey As Long
04207     Dim lRetVal As Long
04208
04209     If (RegCreateKeyEx(m_IKey, SubKey, 0, vbNullString, REG_OPTION_NON_VOLATILE, _
04210         KEY_ALL_ACCESS, 0, hNewKey, lRetVal) = ERROR_SUCCESS) Then
04211         RegCreateKey = hNewKey
04212     Else
04213         RegCreateKey = -1
04214     End If
04215 End Function
04216
04217 Private Function Min(ByVal x As Long, ByVal y As Long) As Long
04218     Min = IIf(x < y, x, y)
04219 End Function
04220
04221

```

```

04222 Option Explicit
04223 ' demo project showing how to create a custom MSDE install
04224 ' by Bryan Stafford of New Vision Software - newvision@myps.org
04225 ' this demo is released into the public domain "as is" without
04226 ' warranty or guaranty of any kind. In other words, use at
04227 ' your own risk.
04228 '
04229 ' Copyright © 2002 Bryan Stafford/New Vision Software
04230 '
04231 ' the code in this demo is copyrighted and may only be used in
04232 ' accordance with the rules set forth in the "House Rules"
04233 ' section of the web page found at http://www.myps.org/vbvision/
04234 '
04235 ' see the general declararions section of MMain.bas for more info
04236 ' on this project
04237
04238 Private Const GWL_WNDPROC As Long = (-4&)
04239
04240
04241 Private Const API_FALSE As Long = 0&
04242 Private Const API_TRUE As Long = 1&
04243
04244
04245 Private Type RECT
04246     Left As Long
04247     Top As Long
04248     Right As Long
04249     Bottom As Long
04250 End Type
04251
04252 Private Declare Function IsWindow Lib "user32" (ByVal hWnd&) As Long
04253
04254 Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (lpDest As Any, lpSource As Any, ByVal
    cBytes&)
04255
04256 Private Declare Function SetProp Lib "user32" Alias "SetPropA" (ByVal hWnd As Long, ByVal lpString As
    String, ByVal hData As Long) As Long
04257 Private Declare Function GetProp Lib "user32" Alias "GetPropA" (ByVal hWnd As Long, ByVal lpString As
    String) As Long
04258 Private Declare Function RemoveProp Lib "user32" Alias "RemovePropA" (ByVal hWnd&, ByVal lpString$) As Long
04259
04260 Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hWnd&, _
    ByVal nIndex&, ByVal dwNewLong&) As Long
04261
04262 Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" (ByVal lpPrevWndFunc, _
    ByVal hWnd&, ByVal MSG&, ByVal wParam&, ByVal lParam&) As Long
04263
04264 Private Declare Function ReleaseCapture Lib "user32" () As Long
04265
04266 Private Enum eDrEdgeConstants
04267     DB_NOFRAME = 0&
04268     DB_SUNKEN = 2&
04269     DB_RAISED = 4&
04270     DB_RAISEDBUTTON = 5&
04271     DB_FRAME = 6&
04272     DB_DUBBLESUNKEN = 10&
04273 End Enum
04274
04275 ' DrawState constants
04276 Private Const DSS_DISABLED As Long = &H20&
04277 Private Const DSS_MONO As Long = &H80&
04278 Private Const DSS_NORMAL As Long = &H0&
04279 Private Const DSS_UNION As Long = &H10&
04280 Private Const DST_BITMAP As Long = &H4&
04281 Private Const DST_COMPLEX As Long = &H0&
04282 Private Const DST_ICON As Long = &H3&
04283 Private Const DST_PFI XTEXT As Long = &H2&
04284 Private Const DST_TEXT As Long = &H1&
04285
04286 Private Enum PICTYPE
04287     PICTYPE_UNINITIALIZED = -1&
04288     PICTYPE_NONE = 0&
04289     PICTYPE_BITMAP = 1&
04290     PICTYPE_METAFILE = 2&
04291     PICTYPE_ICON = 3&
04292     PICTYPE_ENHMETAFILE = 4&
04293 End Enum

```

```

04297
04298 Private Enum PICATTRIBUTES
04299     PICTURE_SCALABLE = 1&
04300     PICTURE_TRANSPARENT = 2&
04301 End Enum
04302
04303 ' alias used to draw graphics, notice the IData parameter
04304 Private Declare Function DrawState Lib "user32" Alias "DrawStateA" (ByVal hdc&, ByVal hBrush&, _
04305     ByVal lpDrawStateProc&, ByVal IData&, ByVal wData&, ByVal x&, ByVal y&, ByVal cx&, _
04306     ByVal cy&, ByVal fFlags&) As Long
04307
04308 Private Declare Function DrawStateText Lib "user32" Alias "DrawStateA" (ByVal hdc&, ByVal hBrush&, _
04309     ByVal lpDrawStateProc&, ByVal IData$, ByVal wData&, ByVal x&, ByVal y&, ByVal cx&, _
04310     ByVal cy&, ByVal fFlags&) As Long
04311
04312
04313 Private Const BF_ADJUST As Long = &H2000&
04314 Private Const BF_BOTTOM As Long = &H8&
04315 Private Const BF_BOTTOMLEFT As Long = &H9&
04316 Private Const BF_BOTTOMRIGHT As Long = &HC&
04317 Private Const BF_DIAGONAL As Long = &H10&
04318 Private Const BF_FLAT As Long = &H4000&
04319 Private Const BF_LEFT As Long = &H1&
04320 Private Const BF_MIDDL E As Long = &H800&
04321 Private Const BF_MONO As Long = &H8000&
04322 Private Const BF_RECT As Long = &HF&
04323 Private Const BF_RIGHT As Long = &H4&
04324 Private Const BF_SOFT As Long = &H1000&
04325 Private Const BF_SOFTRECT As Long = (BF_SOFT Or BF_RECT)
04326 Private Const BF_TOP As Long = &H2&
04327 Private Const BF_TOPLEFT As Long = &H3&
04328 Private Const BF_TOPRIGHT As Long = &H6&
04329
04330 Private Const BDR_INNER As Long = &HC&
04331 Private Const BDR_OUTER As Long = &H3&
04332 Private Const BDR_RAISED As Long = &H5&
04333 Private Const BDR_RAISEDINNER As Long = &H4&
04334 Private Const BDR_RAISEDOUTER As Long = &H1&
04335 Private Const BDR_SUNKEN As Long = &HA&
04336 Private Const BDR_SUNKENINNER As Long = &H8&
04337 Private Const BDR_SUNKENOUTER As Long = &H2&
04338 Private Const EDGE_BUMP As Long = (BDR_RAISEDOUTER Or BDR_SUNKENINNER)
04339 Private Const EDGE_ETCHED As Long = (BDR_SUNKENOUTER Or BDR_RAISEDINNER)
04340 Private Const EDGE_RAISED As Long = (BDR_RAISEDOUTER Or BDR_RAISEDINNER)
04341 Private Const EDGE_SUNKEN As Long = (BDR_SUNKENOUTER Or BDR_SUNKENINNER)
04342
04343 Private Declare Function DrawEdge Lib "user32" (ByVal hdc&, qrc As RECT, ByVal edge&, ByVal grfFlags&) As
04344     Long
04345
04346 Private Declare Function CreateBitmap Lib "gdi32" (ByVal nWidth&, ByVal nHeight&, ByVal nPlanes&, ByVal
04347     nBitsCount&, lpBits As Any) As Long
04348
04349 Private Declare Function CreateCompatibleBitmap Lib "gdi32" (ByVal hdc&, ByVal nWidth&, ByVal nHeight&) As
04350     Long
04351
04352 Private Declare Function CreateCompatibleDC Lib "gdi32" (ByVal hdc&) As Long
04353
04354 Private Declare Function SetBkColor Lib "gdi32" (ByVal hdc&, ByVal crColor&) As Long
04355
04356 Private Declare Function SelectObject Lib "gdi32" (ByVal hdc&, ByVal hObject&) As Long
04357
04358 Private Declare Function DeleteDC Lib "gdi32" (ByVal hdc&) As Long
04359
04360 Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject&) As Long
04361
04362 Private Declare Function GetBkColor Lib "gdi32" (ByVal hdc&) As Long
04363
04364 Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC&, ByVal x&, ByVal y&, ByVal nWidth&, ByVal
04365     nHeight&, ByVal hSrcDC&, ByVal xSrc&, ByVal ySrc&, ByVal dwRop&) As Long
04366
04367 Private Declare Function FillRect Lib "user32" (ByVal hdc&, lpRect As RECT, ByVal hBrush&) As Long
04368
04369 Private Declare Function CreateSolidBrush Lib "gdi32" (ByVal crColor&) As Long
04370
04371 Private Const RASTERCAPS As Long = 38&
04372 Private Const SI ZEPALLETTE As Long = 104&
04373 Private Const RC_PALETTE As Long = &H100&
04374
04375 Private Type PALETTEENTRY
04376     peRed As Byte
04377     peGreen As Byte
04378     peBlue As Byte

```

```

04372 1 peFlags As Byte
04373 End Type
04374
04375 Private Type LOGPALETTE256
04376     palVersion As Integer
04377     palNumEntries As Integer
04378     palPalEntry(255) As PALETTEENTRY
04379 End Type
04380
04381 Private Type BITMAP
04382     bmType As Long
04383     bmWidth As Long
04384     bmHeight As Long
04385     bmWidthBytes As Long
04386     bmPlanes As Integer
04387     bmBitsPixel As Integer
04388     bmBits As Long
04389 End Type
04390
04391 Private Type GUID
04392     Data1 As Long
04393     Data2 As Integer
04394     Data3 As Integer
04395     Data4(7) As Byte
04396 End Type
04397
04398 Private Type PICTDESC_BMP
04399     Size As Long
04400     Type As Long
04401     hBmp As Long
04402     hPal As Long
04403     Reserved As Long
04404 End Type
04405
04406 Private Declare Function GetDeviceCaps Lib "gdi32" (ByVal hdc&, ByVal iCapability&) As Long
04407 Private Declare Function GetSystemPaletteEntries Lib "gdi32" (ByVal hdc&, ByVal wStartIndex&, ByVal
» wNumEntries&, lpPaletteEntries As PALETTEENTRY) As Long
04408 Private Declare Function OleCreatePictureIndirect Lib "olepro32.dll" (PicDesc As PICTDESC_BMP, RefIID As
» GUID, ByVal fPictureOwnsHandle&, IPic As IPicture) As Long
04409 Private Declare Function RealizePalette Lib "gdi32" (ByVal hdc&) As Long
04410 Private Declare Function SelectPalette Lib "gdi32" (ByVal hdc&, ByVal hPalette&, ByVal bForceBackground&)
» As Long
04411 Private Declare Function CreatePalette Lib "gdi32" (lpLogPalette As LOGPALETTE256) As Long
04412
04413
04414
04415 Private m_fDesign As Boolean
04416
04417 Private m_bEnabled As Boolean
04418 Private m_sCaption As String
04419
04420 ' one picture to work with and one to store the picture that is saved to the control's property
04421 Private m_oPic As IPicture
04422 Private m_oSavePic As IPicture
04423
04424 Public Enum CAPTION_ALIGNMENT
04425     pbLeft = 0&
04426     pbTop
04427     pbRight
04428     pbBottom
04429 End Enum
04430
04431 Private m_eAlignment As CAPTION_ALIGNMENT
04432
04433 Event Click()

```

```

04434
04435 Public Property Get Caption() As String
04436     Caption = m_sCaption
04437 End Property

```

```

04438 Public Property Let Caption(ByVal sNewCaption As String)
04439     m_sCaption = sNewCaption
04440     PropertyChanged "Caption"
04441     UserControl.Refresh
04442 End Property

```

```

04443
04444 Public Property Get CaptionAlignment() As CAPTION_ALIGNMENT
04445     CaptionAlignment = m_eAlignment
04446 End Property

```

```

04447 Public Property Let CaptionAlignment(ByVal eAlignment As CAPTION_ALIGNMENT)
04448     m_eAlignment = eAlignment
04449     PropertyChanged "CaptionAlignment"
04450     UserControl.Refresh
04451 End Property
-----
04452
04453 Public Property Get Enabled() As Boolean
04454     Enabled = m_bEnabled
04455 End Property
-----
04456 Public Property Let Enabled(ByVal sNewValue As Boolean)
04457     m_bEnabled = sNewValue
04458     PropertyChanged "Enabled"
04459     UserControl.Refresh
04460 End Property
-----
04461
04462 Public Property Get Font() As Font
04463     Set Font = UserControl.Font
04464 End Property
-----
04465 Public Property Set Font(ByVal New_Font As Font)
04466     Set UserControl.Font = New_Font
04467     PropertyChanged "Font"
04468     UserControl.Refresh
04469 End Property
-----
04470
04471 Public Property Get Picture() As IPictureDisp
04472     Set Picture = m_oSavePic
04473 End Property
-----
04474 Public Property Set Picture(ByVal oPicture As IPictureDisp)
04475     Set m_oPic = oPicture
04476     Set m_oSavePic = oPicture
04477     PropertyChanged "Picture"
04478     UserControl.Refresh
04479 End Property
-----
04480
04481 Private Sub UserControl_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
04482     If (Button = 1) And (m_bEnabled = True) Then DrawButton DB_DUBBLESUNKEN
04483 End Sub
-----
04484
04485 Private Sub UserControl_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
04486     With UserControl
04487         If (Button = 1) And (m_bEnabled = True) Then
04488             If ((x >= False) And (x <= .ScaleWidth)) And ((y >= False) And (y <= .ScaleHeight)) Then
04489                 DrawButton DB_DUBBLESUNKEN
04490             Else
04491                 DrawButton DB_RAISEDBUTTON
04492             End If
04493         Else
04494             DrawButton DB_RAISEDBUTTON
04495         End If
04496     End With
04497 End Sub
-----
04498
04499
04500
04501 Private Sub UserControl_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
04502     With UserControl
04503         If (Button = 1) And (m_bEnabled = True) Then
04504             If (((x >= False) And (x <= .ScaleWidth)) And ((y >= False) And (y <= .ScaleHeight))) Then
04505                 DrawButton DB_RAISEDBUTTON
04506                 RaiseEvent Click
04507             End If
04508         End If
04509     End With
04510 End Sub
-----
04511
04512 Private Sub DrawButton(ByVal fState As Long)
04513     ' draw the button the state is the flag that will be passed to DrawEdge
04514
04515     Dim rectBOX As RECT, nTextLeftPos&, nTextTopPos&, nPicLeftPos&, nPicTopPos&, nPicWidth&, nPicHeight&,
    » hPicture&
04516     Dim nPicAndTextTotalWidth&, nPicAndTextTotalHeight&, fPicType&, fInabledState&
04517     Dim oPic As IPicture
04518
04519     With UserControl

```

```

04520 1 2 ' set the rect for the DrawEdge call
04521 rectBOX.Right = .ScaleWidth
04522 rectBOX.Bottom = .ScaleHeight
04523
04524 If Not m_oPic Is Nothing Then
04525     ' if we have a graphic handle, deal with drawing the picture
04526     If m_oPic.Handle <> 0 Then
04527
04528         ' translate the width and height into pixels
04529         nPicWidth = UserControl.ScaleX(m_oPic.Width, vbMetrics, vbPixels)
04530         nPicHeight = UserControl.ScaleY(m_oPic.Height, vbMetrics, vbPixels)
04531
04532         ' if there is a caption, calculate the positions of the picture and text based on the chosen
04533         ' alignment
04534         If Len(m_sCaption) > 0 Then
04535             nPicAndTextTotalWidth = nPicWidth + .TextWidth(m_sCaption)
04536             nPicAndTextTotalHeight = nPicHeight + .TextHeight(m_sCaption)
04537
04538             Select Case m_eAlignment
04539             Case pbLeft
04540                 nTextLeftPos = (.ScaleWidth \ 2) - (nPicAndTextTotalWidth \ 2)
04541                 nTextTopPos = (.ScaleHeight \ 2) - (.TextHeight(m_sCaption) \ 2)
04542
04543                 nPicLeftPos = nTextLeftPos + .TextWidth(m_sCaption)
04544                 nPicTopPos = (.ScaleHeight \ 2) - (nPicHeight \ 2)
04545
04546             Case pbTop
04547                 nTextLeftPos = (.ScaleWidth \ 2) - (.TextWidth(m_sCaption) \ 2)
04548                 nPicLeftPos = (.ScaleWidth \ 2) - (nPicWidth \ 2)
04549
04550                 nTextTopPos = (.ScaleHeight \ 2) - (nPicAndTextTotalHeight \ 2)
04551                 nPicTopPos = nTextTopPos + .TextHeight(m_sCaption)
04552
04553             Case pbRight
04554                 nPicLeftPos = (.ScaleWidth \ 2) - (nPicAndTextTotalWidth \ 2)
04555                 nTextTopPos = (.ScaleHeight \ 2) - (.TextHeight(m_sCaption) \ 2)
04556
04557                 nTextLeftPos = nPicLeftPos + nPicWidth
04558                 nPicTopPos = (.ScaleHeight \ 2) - (nPicHeight \ 2)
04559
04560             Case pbBottom
04561                 nTextLeftPos = (.ScaleWidth \ 2) - (.TextWidth(m_sCaption) \ 2)
04562                 nPicLeftPos = (.ScaleWidth \ 2) - (nPicWidth \ 2)
04563
04564                 nPicTopPos = (.ScaleHeight \ 2) - (nPicAndTextTotalHeight \ 2)
04565                 nTextTopPos = nPicTopPos + nPicHeight
04566             End Select
04567
04568         Else ' no caption so just center the picture
04569             nPicLeftPos = (.ScaleWidth \ 2) - (nPicWidth \ 2)
04570             nPicTopPos = (.ScaleHeight \ 2) - (nPicHeight \ 2)
04571         End If
04572
04573         ' set the type of graphic to be passed to the DrawState function
04574         Select Case m_oPic.Type
04575         Case PICTURE_BITMAP
04576             fPicType = DST_BITMAP
04577             hPicture = m_oPic.Handle
04578
04579         Case PICTURE_ICON
04580             fPicType = DST_ICON
04581             hPicture = m_oPic.Handle
04582
04583         Case PICTURE_METAFILE, PICTURE_ENHMETAFILE
04584             ' if this is a metafile, we need to convert it to a bitmap for the DrawState call
04585             fPicType = DST_BITMAP
04586
04587             Dim hDCMemory&, hBmp&, hBmpPrev&
04588
04589             ' create the DC and bitmap we will use
04590             hDCMemory = CreateCompatibleDC(UserControl.hDC)
04591             hBmp = CreateCompatibleBitmap(UserControl.hDC, nPicWidth, nPicHeight)
04592             hBmpPrev = SelectObject(hDCMemory, hBmp)
04593
04594             m_oPic.Render hDCMemory, 0, 0, nPicWidth, nPicHeight, 0, 0, m_oPic.Width, m_oPic.Height, ByVal 0&
04595
04596         If (m_oPic.Attributes And PICTURE_TRANSPARENT) > 0 Then

```

```

04597 1 2 3 4 5 6 Dim hTransDC&, hPrevTransBmp&, nColor&, hBrush&, hPrevBrush&, rc As RECT
04598
04599 ' create the DC and bitmap we will use
04600 hTransDC = CreateCompatibleDC(UserControl.hDC)
04601 hBmp = CreateCompatibleBitmap(UserControl.hDC, nPicWidth, nPicHeight)
04602 hPrevTransBmp = SelectObject(hTransDC, hBmp)
04603
04604 nColor = GetBkColor(UserControl.hDC)
04605 Call SetBkColor(hTransDC, nColor)
04606
04607 hBrush = CreateSolidBrush(nColor)
04608
04609 hPrevBrush = SelectObject(hTransDC, hBrush)
04610
04611 rc.Right = nPicWidth
04612 rc.Bottom = nPicHeight
04613
04614 Call FillRect(hTransDC, rc, hBrush)
04615
04616 TransparentBlt hTransDC, rc, hDCMemory, rc, 0&
04617
04618 ' once we convert the metafile to a bitmap, we then save the bitmap back to
04619 ' the picture object so that we only have to convert it one time
04620 Set m_oPic = PictureFromDC(hTransDC, 0&, 0&, nPicWidth, nPicHeight)
04621
04622 hPicture = m_oPic.Handle
04623
04624 ' clean up
04625 Call DeleteObject(SelectObject(hTransDC, hPrevBrush))
04626
04627 Call DeleteObject(SelectObject(hTransDC, hPrevTransBmp))
04628
04629 DeleteDC hTransDC
04630
04631 } Else
04632 ' get the bitmap from the DC
04633 hPicture = SelectObject(hDCMemory, hBmpPrev)
04634
04635 ' once we convert the metafile to a bitmap, we then save the bitmap back to
04636 ' the picture object so that we only have to convert it one time
04637 Set m_oPic = PictureFromDC(hDCMemory, 0&, 0&, nPicWidth, nPicHeight)
04638
04639 Call DeleteObject(SelectObject(hDCMemory, hBmpPrev))
04640 } End If
04641
04642 ' clean up
04643 DeleteDC hDCMemory
04644
04645 } End Select
04646
04647 } Else
04648 ' get the current x & y for printing the text
04649 nTextLeftPos = (.ScaleWidth \ 2) - (.TextWidth(m_sCaption) \ 2)
04650 nTextTopPos = (.ScaleHeight \ 2) - (.TextHeight(m_sCaption) \ 2)
04651 } End If
04652 } Else
04653 ' get the current x & y for printing the text
04654 nTextLeftPos = (.ScaleWidth \ 2) - (.TextWidth(m_sCaption) \ 2)
04655 nTextTopPos = (.ScaleHeight \ 2) - (.TextHeight(m_sCaption) \ 2)
04656 } End If
04657
04658 ' determine the enabled state to draw
04659 If m_bEnabled = False Then
04660 fState = DB_RAISEDBUTTON
04661 fEnabledState = DSS_DISABLED
04662
04663 } Else
04664 ' if it's pressed, offset the text to the right and down
04665 If fState = DB_DUBBLESUNKEN Then
04666 nTextLeftPos = nTextLeftPos + 1
04667 nTextTopPos = nTextTopPos + 1
04668
04669 nPicLeftPos = nPicLeftPos + 1
04670 nPicTopPos = nPicTopPos + 1
04671 } End If
04672 } End If
04673
04674 ' clear what's already there
04675 .Cls

```

```

04676 1 2
04677     ' if we have a picture, draw it
04678     If (hPicture <> 0) Then Call DrawState(UserControl.hDC, 0&, 0&, hPicture, 0&, nPicLeftPos, nPicTopPos,
>       nPicWidth, nPicHeight, fPicType Or fEnabledState)
04679
04680     ' print the text
04681     Call DrawStateText(.hDC, 0&, 0&, m_sCaption, Len(m_sCaption), nTextLeftPos, _
04682       nTextTopPos, 0&, 0&, DST_PREFIXTEXT Or fEnabledState)
04683
04684     ' draw the edge of the button
04685     If (fState <> 0) And (m_bEnabled = True) Then Call DrawEdge(.hDC, rectBOX, fState, BF_SOFTRECT)
04686     End With
04687
04688 End Sub

```

```

04689
04690 Private Sub UserControl_AmbientChanged(PropertyName As String)
04691     If m_fDesign Then
04692         If PropertyName = "DisplayName" Then UserControl.Refresh
04693     End If
04694 End Sub

```

```

04695
04696 Private Sub UserControl_InitProperties()
04697     m_bEnabled = True
04698     SetDesignMode
04699     If Not m_fDesign Then
04700         UserControl.AutoRedraw = True
04701     End If
04702 End Sub

```

```

04703
04704 Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
04705     m_bEnabled = PropBag.ReadProperty("Enabled", True)
04706     m_sCaption = PropBag.ReadProperty("Caption", vbNullString)
04707     Set UserControl.Font = PropBag.ReadProperty("Font", Ambient.Font)
04708     Set m_oSavePic = PropBag.ReadProperty("Picture", Nothing)
04709     m_eAlignment = PropBag.ReadProperty("CaptionAlignment", pbLeft)
04710
04711     Set m_oPic = m_oSavePic
04712
04713     SetDesignMode
04714     If Not m_fDesign Then
04715         UserControl.AutoRedraw = True
04716     End If
04717 End Sub

```

```

04718
04719 Private Sub UserControl_Show()
04720     If Not m_fDesign Then DrawButton DB_RAISED_BUTTON
04721 End Sub

```

```

04722
04723 Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
04724     PropBag.WriteProperty "Enabled", m_bEnabled, True
04725     PropBag.WriteProperty "Caption", m_sCaption, vbNullString
04726     PropBag.WriteProperty "Font", UserControl.Font, Ambient.Font
04727     PropBag.WriteProperty "Picture", m_oSavePic, Nothing
04728     PropBag.WriteProperty "CaptionAlignment", m_eAlignment, pbLeft
04729 End Sub

```

```

04730
04731 Private Sub UserControl_Paint()
04732     DrawButton DB_RAISED_BUTTON
04733 End Sub

```

```

04734
04735 Private Sub NoPropertySheet()
04736     ' To keep a property out of both the locals window and the
04737     ' property sheet, check the 'Don't show in Property Browser'
04738     ' attribute, available on the Advanced tab of the Tools/Procedure
04739     ' Attributes dialog. To stop the item from showing in the
04740     ' property sheet, but still support it in the locals window,
04741     ' use this approach.
04742
04743     ' If you stop on this error, then right click on the
04744     ' code pane and choose Toggle/Break on Unhandled Errors.
04745     ' You can set the default for this setting in the Tools/Options
04746     ' dialog on the General tab.
04747     If m_fDesign Then Err.Raise 394 'GetNotSupported
04748 End Sub

```

```

04749

```

```

04750 Private Sub DesignTimeOnly()
04751     If Not m_fDesign Then Err.Raise 382
04752 End Sub

```

```

04753 Private Sub SetDesignMode()
04754     On Error Resume Next
04755     m_fDesign = Not Ambient.UserMode
04756     If Err Then m_fDesign = True
04757     Err.Clear
04758 End Sub

```

```

04760 Private Function PictureFromDC(ByVal hDCSrc&, ByVal nLeft&, ByVal nTop&, ByVal nWidth&, ByVal nHeight&) As
04761     StdPicture
    »
04762     Dim hDCMemory&, hBmp&, hBmpPrev&, hPal&, hPalPrev&
04763     Dim fHasPalette&, nPaletteEntries&, LogPal As LOGPALETTE256
04764
04765     ' create the DC and bitmap we will use
04766     hDCMemory = CreateCompatibleDC(hDCSrc)
04767     hBmp = CreateCompatibleBitmap(hDCSrc, nWidth, nHeight)
04768     hBmpPrev = SelectObject(hDCMemory, hBmp)
04769
04770     ' determine whether or not the video supports 256 color palettes
04771     nPaletteEntries = GetDeviceCaps(hDCSrc, SI_ZEPALLETTE)
04772     fHasPalette = GetDeviceCaps(hDCSrc, RASTERCAPS) And RC_PALETTE
04773
04774     ' if this is 256 color video, we need to create and add a palette to our DC
04775     If fHasPalette And (nPaletteEntries = 256) Then
04776         LogPal.palVersion = &H300
04777         LogPal.palNumEntries = 256
04778         GetSystemPaletteEntries hDCSrc, 0, 256, LogPal.palPalEntry(0)
04779         hPal = CreatePalette(LogPal)
04780         hPalPrev = SelectPalette(hDCMemory, hPal, 0)
04781         RealizePalette hDCMemory
04782     End If
04783
04784     ' copy the passed image into the local DC
04785     BitBlt hDCMemory, 0, 0, nWidth, nHeight, hDCSrc, nLeft, nTop, vbSrcCopy
04786
04787     ' get the bitmap from the DC
04788     hBmp = SelectObject(hDCMemory, hBmpPrev)
04789
04790     ' if we created a palette, get it from the DC
04791     If fHasPalette And (nPaletteEntries = 256) Then
04792         hPal = SelectPalette(hDCMemory, hPalPrev, 0)
04793     End If
04794
04795     ' clean up
04796     DeleteDC hDCMemory
04797
04798     ' create a picture from the bitmap and return it to the caller
04799     Set PictureFromDC = PictureFromBitmap(hBmp, hPal)
04800 End Function

```

```

04803 Private Function PictureFromBitmap(ByVal hBmp&, ByVal hPal&) As StdPicture
04804
04805     Dim IPictureIID As GUID, IPic As IPicture, tagPic As PICTDESC_BMP
04806     Dim lpGUID&
04807
04808     ' fill in the IPicture GUID {7BF80980-BF32-101A-8BBB-00AA00300CAB}
04809     With IPictureIID
04810         .Data1 = &H7BF80980
04811         .Data2 = &HBF32
04812         .Data3 = &H101A
04813         .Data4(0) = &H8B
04814         .Data4(1) = &HBB
04815         .Data4(2) = &HO
04816         .Data4(3) = &HAA
04817         .Data4(4) = &HO
04818         .Data4(5) = &H30
04819         .Data4(6) = &HC
04820         .Data4(7) = &HAB
04821     End With
04822
04823     ' set the properties on the picture object
04824     With tagPic
04825         1 2

```

```

04827 1 2 .Size = Len(tagPic)
04828 .Type = vbPicTypeBitmap
04829 .hBmp = hBmp
04830 .hPal = hPal
04831 End With
04832
04833 ' create a picture that will delete it's bitmap when it is finished with it
04834 OleCreatePictureIndirect tagPic, IPictureID, API_TRUE, IPic
04835
04836 ' return the picture to the caller
04837 Set PictureFromBitmap = IPic
04838
04839 End Function

```

```

04840
04841 Private Sub TransparentBit(hDestDC&, lpDestRect As RECT, hSrcDC&, lpSrcRect As RECT, ByVal TransColor&)
04842
04843 Dim hInvDC&, hMaskDC&, hResultDC&, hInvBmp&, hMaskBmp&, hResultBmp&
04844 Dim hInvPrevBmp&, hMaskPrevBmp&, hDestPrevBmp&, nSrcWidth&, nSrcHeight&, nOriginalColor&
04845
04846 With lpSrcRect
04847     nSrcWidth = .Right - .Left
04848     nSrcHeight = .Bottom - .Top
04849 End With
04850
04851 ' create the mask and invert stage DCs and bitmaps
04852 hInvDC = CreateCompatibleDC(hDestDC)
04853 hMaskDC = CreateCompatibleDC(hDestDC)
04854 ' monochrome bitmaps for the masks
04855 hInvBmp = CreateBitmap(nSrcWidth, nSrcHeight, 1, 1, ByVal 0&)
04856 hMaskBmp = CreateBitmap(nSrcWidth, nSrcHeight, 1, 1, ByVal 0&)
04857 hInvPrevBmp = SelectObject(hInvDC, hInvBmp)
04858 hMaskPrevBmp = SelectObject(hMaskDC, hMaskBmp)
04859
04860 ' create the DC and bitmap to hold the result
04861 hResultDC = CreateCompatibleDC(hDestDC)
04862 ' color bitmap for final result
04863 hResultBmp = CreateCompatibleBitmap(hDestDC, nSrcWidth, nSrcHeight)
04864 hDestPrevBmp = SelectObject(hResultDC, hResultBmp)
04865
04866 ' create mask: set background color of source to transparent color.
04867 nOriginalColor = SetBkColor(hSrcDC, TransColor)
04868 With lpSrcRect
04869     Call BitBlt(hMaskDC, 0, 0, nSrcWidth, nSrcHeight, hSrcDC, .Left, .Top, vbSrcCopy)
04870 End With
04871 TransColor = SetBkColor(hSrcDC, nOriginalColor)
04872
04873 ' create inverse of mask to AND w/ source & combine w/ background.
04874 Call BitBlt(hInvDC, 0, 0, nSrcWidth, nSrcHeight, hMaskDC, 0, 0, vbNotSrcCopy)
04875
04876 ' copy background bitmap to result & create final transparent bitmap
04877 With lpDestRect
04878     Call BitBlt(hResultDC, 0, 0, nSrcWidth, nSrcHeight, hDestDC, .Left, .Top, vbSrcCopy)
04879
04880     ' AND mask bitmap w/ result DC to punch hole in the background by
04881     ' painting black area for non-transparent portion of source bitmap.
04882     Call BitBlt(hResultDC, 0, 0, nSrcWidth, nSrcHeight, hMaskDC, 0, 0, vbSrcAnd)
04883
04884     ' AND inverse mask w/ source bitmap to turn off bits associated
04885     ' with transparent area of source bitmap by making it black.
04886     Call BitBlt(hSrcDC, 0, 0, nSrcWidth, nSrcHeight, hInvDC, 0, 0, vbSrcAnd)
04887
04888     ' XOR result w/ source bitmap to make background show through.
04889     Call BitBlt(hResultDC, 0, 0, nSrcWidth, nSrcHeight, hSrcDC, 0, 0, vbSrcPaint)
04890
04891     ' copy result to the dest DC
04892     Call BitBlt(hDestDC, .Left, .Top, nSrcWidth, nSrcHeight, hResultDC, 0, 0, vbSrcCopy)
04893 End With
04894
04895 ' clean up after ourselves
04896 Call DeleteObject(SelectObject(hMaskDC, hMaskPrevBmp))
04897 Call DeleteObject(SelectObject(hInvDC, hInvPrevBmp))
04898 Call DeleteObject(SelectObject(hResultDC, hDestPrevBmp))
04899
04900 DeleteDC hMaskDC
04901 DeleteDC hInvDC
04902 DeleteDC hResultDC
04903
04904 End Sub
04905

```

```

04906 Option Explicit
04907 ' demo project showing how to create a custom MSDE install
04908 ' by Bryan Stafford of New Vision Software - newvision@myps.org
04909 ' this demo is released into the public domain "as is" without
04910 ' warranty or guaranty of any kind. In other words, use at
04911 ' your own risk.
04912 '
04913 ' Copyright © 2002 Bryan Stafford/New Vision Software
04914 '
04915 ' the code in this demo is copyrighted and may only be used in
04916 ' accordance with the rules set forth in the "House Rules"
04917 ' section of the web page found at http://www.myps.org/vbvision/
04918 '
04919 ' see the general declararions section of MMain.bas for more info
04920 ' on this project
04921
04922
04923 Public Enum TEXT_LOCATION
04924     Center = 0&
04925     [Left Align] = 1&
04926     [Right Align] = 2&
04927 End Enum
04928
04929 Private m_eTextAligment As TEXT_LOCATION
04930
04931
04932 Public Enum FLOOD_STYLE
04933     [No Text] = 0&
04934     [Text Only] = 1&
04935     Both = 2&
04936 End Enum
04937
04938 Private m_eFloodStyle As FLOOD_STYLE
04939
04940 Private m_zInvertedTextColor As OLE_COLOR
04941
04942 Private m_fDesign As Boolean
04943
04944 Private m_nMaxPercentage As Long
04945 Private m_nMinPercentage As Long
04946
04947 Private m_nProgressUnits As Long
04948 Private m_nCurrentPercentage As Long
04949 Private m_nCurrentUnit As Long
04950
04951 Private m_dblStartTime As Double
04952
04953 Public Property Get Font() As Font
04954     Set Font = UserControl.Font
04955 End Property
04956
04957 Public Property Set Font(ByVal New_Font As Font)
04958     Set UserControl.Font = New_Font
04959     PropertyChanged "Font"
04960     UserControl.Refresh
04961 End Property
04962
04963 Public Property Get EstamatedTimeToFinish() As Double
04964     EstamatedTimeToFinish = ((Now - m_dblStartTime) / (m_nCurrentUnit + 1)) * (m_nProgressUnits -
»     m_nCurrentUnit)
04965 End Property
04966
04967 Public Property Get CurrentPercentage() As Long
04968     CurrentPercentage = m_nCurrentPercentage
04969 End Property
04970
04971 Public Property Get CurrentUnit() As Long
04972     CurrentUnit = m_nCurrentUnit
04973 End Property
04974
04975 Public Property Get ProgressUnits() As Long
04976     ProgressUnits = m_nProgressUnits
04977 End Property
04978
04979 Public Property Let ProgressUnits(ByVal nNewVal &)
04980     m_nProgressUnits = nNewVal
04981 End Property

```

```

04980
04981 Public Property Get FloodStyle() As FLOOD_STYLE
04982     FloodStyle = m_eFloodStyle
04983 End Property
-----
04984 Public Property Let FloodStyle(ByVal nNewVal As FLOOD_STYLE)
04985     m_eFloodStyle = nNewVal
04986 End Property
-----
04987
04988 Public Property Get TextAlignment() As TEXT_LOCATION
04989     TextAlignment = m_eTextAlignment
04990 End Property
-----
04991 Public Property Let TextAlignment(ByVal nNewVal As TEXT_LOCATION)
04992     m_eTextAlignment = nNewVal
04993 End Property
-----
04994
04995 Public Property Get MaxPercentage() As Long
04996     MaxPercentage = m_nMaxPercentage
04997 End Property
-----
04998 Public Property Let MaxPercentage(ByVal nNewVal &)
04999     m_nMaxPercentage = nNewVal
05000 End Property
-----
05001
05002 Public Property Get MinPercentage() As Long
05003     MinPercentage = m_nMinPercentage
05004 End Property
-----
05005 Public Property Let MinPercentage(ByVal nNewVal &)
05006     m_nMinPercentage = nNewVal
05007 End Property
-----
05008
05009 Public Property Get BackColor() As OLE_COLOR
05010     BackColor = UserControl.BackColor
05011 End Property
-----
05012 Public Property Let BackColor(ByVal oNewVal As OLE_COLOR)
05013     UserControl.BackColor = oNewVal
05014     UserControl.Refresh
05015 End Property
-----
05016
05017 Public Property Get FillColor() As OLE_COLOR
05018     FillColor = UserControl.ForeColor
05019 End Property
-----
05020 Public Property Let FillColor(ByVal oNewVal As OLE_COLOR)
05021     UserControl.ForeColor = oNewVal
05022     UserControl.Refresh
05023 End Property
-----
05024
05025 Public Property Get InvertedTextColor() As OLE_COLOR
05026     InvertedTextColor = m_zInvertedTextColor
05027 End Property
-----
05028 Public Property Let InvertedTextColor(ByVal oNewVal As OLE_COLOR)
05029     m_zInvertedTextColor = oNewVal
05030 End Property
-----
05031
05032 Private Sub SetDesignMode()
05033     On Error Resume Next
05034     m_fDesign = Not Ambient.UserMode
05035     If Err Then m_fDesign = True
05036     Err.Clear
05037 End Sub
-----
05038
05039 Private Sub UserControl_AmbientChanged(PropertyName As String)
05040     If m_fDesign Then
05041         If PropertyName = "DisplayName" Then UserControl.Refresh
05042     End If
05043 End Sub
-----
05044
05045 Private Sub UserControl_InitProperties()
05046     m_nMaxPercentage = 100
05047     UserControl.ForeColor = vbHighLight
05048     UserControl.BackColor = vbButtonFace
05049     m_zInvertedTextColor = vbHighLightText
05050     m_eTextAlignment = Center

```

```

05051 1 m_eFloodStyle = Both
05052
05053     SetDesignMode
05054     'If Not m_fDesign Then UserControl.AutoRedraw = True
05055 End Sub

```

```

05056
05057 Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
05058     Set UserControl.Font = PropBag.ReadProperty("Font", Ambient.Font)
05059     m_nMaxPercentage = PropBag.ReadProperty("MaxPercentage", 100)
05060     m_nMinPercentage = PropBag.ReadProperty("MinPercentage", 0)
05061     UserControl.ForeColor = PropBag.ReadProperty("FillColor", vbHighLight)
05062     UserControl.BackColor = PropBag.ReadProperty("BackColor", vbButtonFace)
05063     m_zInvertedTextColor = PropBag.ReadProperty("InvertedTextColor", vbHighLightText)
05064     m_eTextAlignment = PropBag.ReadProperty("TextAlignment", Center)
05065     m_eFloodStyle = PropBag.ReadProperty("FloodStyle", Both)
05066     m_nProgressUnits = PropBag.ReadProperty("ProgressUnits", 0)
05067
05068     SetDesignMode
05069     'If Not m_fDesign Then UserControl.AutoRedraw = True
05070 End Sub

```

```

05071
05072 Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
05073     PropBag.WriteProperty "Font", UserControl.Font, Ambient.Font
05074     PropBag.WriteProperty "MaxPercentage", m_nMaxPercentage, 100
05075     PropBag.WriteProperty "MinPercentage", m_nMinPercentage, 0
05076     PropBag.WriteProperty "FillColor", UserControl.ForeColor, vbHighLight
05077     PropBag.WriteProperty "BackColor", UserControl.BackColor, vbButtonFace
05078     PropBag.WriteProperty "InvertedTextColor", m_zInvertedTextColor, vbHighLightText
05079     PropBag.WriteProperty "TextAlignment", m_eTextAlignment, Center
05080     PropBag.WriteProperty "FloodStyle", m_eFloodStyle, Both
05081     PropBag.WriteProperty "ProgressUnits", m_nProgressUnits, 0
05082 End Sub

```

```

05083
05084 Private Sub UserControl_Paint()
05085     If m_fDesign Then
05086         With UserControl
05087             .CurrentX = 2
05088             .CurrentY = (.ScaleHeight \ 2) - (.TextHeight("U") \ 2)
05089             If .CurrentY < 2 Then .CurrentY = 2
05090
05091             Print Extender.Name
05092         End With
05093     End If
05094 End Sub

```

```

05095
05096 Public Sub IncrementProgress(Optional ByVal bStartFresh As Boolean)
05097     'this is the method that draws the flood bar
05098
05099     Dim sText$, iFillLen&, nPrintX&, nPrintY&, nSaveColor As OLE_COLOR
05100
05101     If bStartFresh Then
05102         m_dblStartTime = Now
05103         m_nCurrentUnit = 0
05104         m_nCurrentPercentage = m_nMinPercentage
05105     Else
05106         m_nCurrentUnit = m_nCurrentUnit + 1
05107     End If
05108
05109     With UserControl
05110         .Cls
05111         'figure the percentage of completion
05112         If m_nProgressUnits Then iFillLen = ((.ScaleWidth / m_nProgressUnits) * m_nCurrentUnit)
05113
05114         If iFillLen > .ScaleWidth Then
05115             iFillLen = .ScaleWidth
05116             m_nCurrentPercentage = m_nMaxPercentage
05117         Else
05118             m_nCurrentPercentage = CLng(((iFillLen) / .ScaleWidth) * (m_nMaxPercentage - m_nMinPercentage))
05119         End If
05120
05121         'get the string to print
05122         Select Case FloodStyle
05123         Case [Text Only], Both
05124             sText = CStr(m_nCurrentPercentage) & " %"
05125
05126             'set the justification and print the text
05127

```

```

05128 1 2 3 . CurrentY = (. Scal eHei ght - .TextHei ght(sText)) \ 2
05129 If . CurrentY < 0 Then . CurrentY = 0
05130
05131 nPrintY = . CurrentY
05132
05133 { Select Case TextAl ignment
05134 { Case [Left Al ighn]: . CurrentX = 3 'give it a little clear space
05135 { Case [Right Al ighn]: . CurrentX = (. Scal eWi dth - .TextWi dth(sText)) - 6
05136 { Case Center: . CurrentX = (. Scal eWi dth - .TextWi dth(sText)) \ 2
05137 } End Select
05138
05139 nPrintX = . CurrentX
05140
05141
05142 Dim hMemoryDC&, hMemoryBi tmap&, hPrevMemoryBi tmap&, nBackCol or&, nTextCol or&
05143 Dim oFont As IFont, hPrevFont&, hBrush&, hPrevBrush&, nPrevBackMode&, nPrevTextCol or&
05144 Dim rc As RECT
05145
05146 ' create a memory DC and draw the text using the NON-inverted color on it
05147 hMemoryDC = CreateCompati bl eDC(. hDC)
05148
05149 Call SaveDC(hMemoryDC)
05150
05151 hMemoryBi tmap = CreateCompati bl eBi tmap(. hDC, . Scal eWi dth, . Scal eHei ght)
05152
05153 hPrevMemoryBi tmap = Sel ectObj ect(hMemoryDC, hMemoryBi tmap)
05154
05155 { If . BackCol or And &H80000000 Then
05156 { nBackCol or = GetSysCol or(. BackCol or And &HFFFFFF)
05157 { Else
05158 { nBackCol or = . BackCol or
05159 { End If
05160
05161 hBrush = CreateSol idBrush(nBackCol or)
05162
05163 rc. Ri ght = . Scal eWi dth
05164 rc. Bot tom = . Scal eHei ght
05165
05166 Call Fi llRect(hMemoryDC, rc, hBrush)
05167
05168 Call Del eteObj ect(hBrush)
05169
05170 { If . ForeCol or And &H80000000 Then
05171 { nTextCol or = GetSysCol or(. ForeCol or And &HFFFFFF)
05172 { Else
05173 { nTextCol or = . ForeCol or
05174 { End If
05175
05176 nPrevBackMode = SetBkMode(hMemoryDC, vbTransparent)
05177 nPrevTextCol or = SetTextCol or(hMemoryDC, nTextCol or)
05178
05179 Set oFont = . Font
05180 hPrevFont = Sel ectObj ect(hMemoryDC, oFont. hFont)
05181
05182 Call TextOut(hMemoryDC, nPrintX, nPrintY, sText, Len(sText))
05183
05184 Call Sel ectObj ect(hMemoryDC, hPrevFont)
05185 Set oFont = Nothing
05186 Call SetTextCol or(hMemoryDC, nPrevTextCol or)
05187 Call SetBkMode(hMemoryDC, nPrevBackMode)
05188 ' end memory DC drawi ng
05189
05190
05191 ' draw the bar
05192 UserControl. Li ne (0, 0)-(IFi llLen, . Scal eHei ght), , BF
05193
05194 ' change the text color to the inverted color and draw the text
05195 nSaveCol or = . ForeCol or
05196 . ForeCol or = Me. InvertedTextCol or
05197
05198 . CurrentX = nPrintX: . CurrentY = nPrintY
05199
05200 UserControl. Pri nt sText
05201
05202 . ForeCol or = nSaveCol or
05203
05204 ' call Bi tBl it to copy the right side of the bar onto the usercontrol from the memory DC
05205 Call Bi tBl it(. hDC, IFi llLen + 1, 0, . Scal eWi dth, . Scal eHei ght, hMemoryDC, IFi llLen + 1, 0, vbSrcCopy)
05206
1 2 3

```

```
05207 1 2 3 ' delete the memory DC
05208 DeleteObject SelectObject(hMemoryDC, hPrevMemoryBitmap)
05209 RestoreDC hMemoryDC, (-1&)
05210 DeleteDC hMemoryDC
05211
05212
05213 } Case Else
05214     UserControl.Line (0, 0)-(FillLen, ScaleHeight), , BF
05215 }
05216 End Select
05217
05218 End With
05219
05220 End Sub


---


05221
05222 Public Sub Clear()
05223
05224 End Sub
05225
```

Procedure Index for Project - PMSDEInstaller

Pg #	Name	Referenced on Pages	Scope	Type
1	frmConfigureMSDE	2, 45, 47		Form
1	cmdNav_Click	4, 6, 13, 17, 20	Private	Sub
2	ConfigurationCallback	54-56	Public	Sub
1	DisplayDialog	3, 4, 6, 13, 17, 20, 46, 47	Public	Function
2	Form_KeyDown	5, 7, 14, 18, 21	Private	Sub
2	Form_QueryUnload	5, 7, 13, 18, 21	Private	Sub
2	Form_Unload	5, 7, 14, 18, 21	Private	Sub
2	tmrConfigureServer_Timer		Private	Sub
3	frmGetServerInfo	4, 5, 45-47		Form
5	ckNTSecurity_Click		Private	Sub
4	cmdNav_Click	1, 6, 13, 17, 20	Private	Sub
3	DisplayDialog	1, 4, 6, 13, 17, 20, 46, 47	Public	Function
5	Form_KeyDown	2, 7, 14, 18, 21	Private	Sub
5	Form_QueryUnload	2, 7, 13, 18, 21	Private	Sub
5	Form_Unload	2, 7, 14, 18, 21	Private	Sub
6	frmInstallMSDE	7-12, 45, 47		Form
6	cmdNav_Click	1, 4, 13, 17, 20	Private	Sub
12	DisableCancelButton	10	Private	Function
6	DisplayDialog	1, 3, 4, 13, 17, 20, 46, 47	Public	Function
7	Form_KeyDown	2, 5, 14, 18, 21	Private	Sub
7	Form_QueryUnload	2, 5, 13, 18, 21	Private	Sub
7	Form_Unload	2, 5, 14, 18, 21	Private	Sub
7	GetAvailableMSIPackageName	8, 9	Private	Function
9	GetMSIFromProductCode	8	Private	Function
12	ProcessIDFromhWnd		Private	Function
9	RemoveMSIFileNameFromArray	8	Private	Sub
9	tmrStartInstall_Timer		Private	Sub
13	frmInstallMSI	14-16, 45, 46		Form
14	cmdInstallMSI_Click		Private	Sub
13	cmdNav_Click	1, 4, 6, 17, 20	Private	Sub
13	DisplayDialog	1, 3, 4, 6, 17, 20, 46, 47	Public	Function
14	Form_KeyDown	2, 5, 7, 18, 21	Private	Sub
13	Form_QueryUnload	2, 5, 7, 18, 21	Private	Sub
13	Form_Unload	2, 5, 7, 14, 18, 21	Private	Sub
15	InstallMSI	13, 14, 16, 45, 46	Public	Function
14	tmrInstallMSI_Timer		Private	Sub
14	tmrProgress_Timer		Private	Sub
17	frmReadyToInstallMSDE	18, 45, 47		Form
17	cmdNav_Click	1, 4, 6, 13, 20	Private	Sub
17	DisplayDialog	1, 3, 4, 6, 13, 20, 46, 47	Public	Function
18	Form_KeyDown	2, 5, 7, 14, 21	Private	Sub
18	Form_QueryUnload	2, 5, 7, 13, 21	Private	Sub
18	Form_Unload	2, 5, 7, 14, 21	Private	Sub
19	frmTaskbarProxy	45, 47		Form
20	frmWelcome	21, 45, 46		Form
20	cmdNav_Click	1, 4, 6, 13, 17	Private	Sub
20	DisplayDialog	1, 3, 4, 6, 13, 17, 46, 47	Public	Function
21	Form_KeyDown	2, 5, 7, 14, 18	Private	Sub
21	Form_QueryUnload	2, 5, 7, 13, 18	Private	Sub
21	Form_Unload	2, 5, 7, 14, 18	Private	Sub
22	MAPI_Declares	23-29		Module
30	MGeneralUtilities	31-40		Module
37	AddBackslashToPath	9, 10, 15, 38, 41, 52	Public	Function
36	Callwcsrchr	35	Public	Function
35	CompareFileVersions		Public	Function
37	CreateTempFileName	38	Public	Function
32	CustomFormat	33	Public	Function
37	DriveExists		Private	Function
37	FileExists	9, 10, 38, 54, 55	Public	Function
34	GetFileVersion	30, 35, 39	Public	Function
34	GetOSVersion	3, 15, 54	Public	Function
31	HIWORD		Public	Function
30	INDEXTOSTATEIMAGEMASK	31	Public	Function
39	Is2KShell		Public	Function
31	IsArrayEmpty		Public	Function
38	IsCallExported	52	Public	Function
31	LOWORD		Public	Function
31	MAKELONG		Public	Function
31	MAKELPARAM		Public	Function
31	MAKEPOINT		Public	Function

Pg #	Name	Referenced on Pages	Scope	Type
31	Max	3, 8-11, 14, 15, 22-24, 26, 32, 37, 58-60, 71-73	Public	Function
31	Min	6-17, 22-24, 26-28, 34, 36, 39, 42, 44-47, 50-52, 55, 57, 59-61, 67, 69, 71-73	Public	Function
32	NumComp		Public	Function
40	PointerToDWord		Public	Function
39	PointerToStringA		Public	Function
39	PointerToStringW	40	Public	Function
36	RemoveFileNameFromPath		Public	Function
39	RemoveIllegalFileChars		Public	Function
32	Replace	25, 39, 44	Public	Function
37	ReturnComputerName	54	Public	Function
36	ReturnFileExtension		Public	Function
35	ReturnFileName	36, 55	Public	Function
38	ReturnFileTime		Public	Function
32	ReturnModuleEXEName		Public	Function
35	ReturnPathRoot		Public	Function
38	ReturnProcAddress		Public	Function
36	ReturnShortPathName	9, 15, 37, 52, 54	Public	Function
32	StringComp		Public	Function
31	StripLfs	32	Public	Function
31	StripNulls	8-10, 32, 37	Public	Function
33	TransparentBit	42, 67, 70	Public	Sub
37	TrueFalseToYesNo		Public	Function
38	WinTimeToVBATime	39	Public	Function
32	Yeild	10, 15, 54, 55	Public	Sub
41	MHelperFunctions	42, 43		Module
41	AskAboutCancel	1, 4, 7, 13, 17, 20	Public	Function
41	CreateTitlePicture	1, 3, 20, 42	Public	Function
43	PictureFromBitmap	69, 70	Private	Function
42	PictureFromDC	43, 67, 69	Private	Function
43	RunningInIde	41	Public	Function
43	TestIDE		Private	Function
44	MMain	1, 3, 6, 13, 17, 19, 20, 22, 30, 41, 45-57, 62, 71		Module
50	AddLoginToServerRoll	55	Public	Sub
50	AddSQLServerLogin	55	Public	Sub
50	AddUserToDatabase		Public	Sub
49	ChangePassword	55, 56	Public	Sub
49	ChangePasswordNTAuth	56	Public	Sub
53	ConfigureSQLServer	2, 56	Public	Function
52	DBNameIsOK		Public	Function
51	DoesSQLServerExist		Public	Function
56	Encrypt	46, 48, 53	Public	Function
51	GetSQLServerRollName	52, 55	Public	Function
51	GetSQLServerVersion		Public	Function
52	IsMSINeeded	46	Private	Function
51	IsSAPasswordOK		Private	Function
45	Main	1, 3, 6, 13, 17, 19, 20, 22, 27, 30, 41, 44, 46-57, 62, 71	Private	Sub
49	ParseServerInfo	48	Private	Function
48	ProcessCommandline	46, 47, 49	Private	Function
53	RemoveCommandLineKey	48	Public	Sub
53	RemoveRunKey	49	Public	Sub
53	SetCommandLineKey	11	Public	Sub
52	SetRunKey	11, 14	Public	Sub
45	TaskbarProxy	1, 3, 6, 13, 17, 19, 20, 47	Public	Property Get
57	CRegistry	4, 8, 48, 52, 53, 58-61		Class
58	Class_Initialize		Private	Sub
58	DefaultValueType		Public	Property Get
58	DefaultValueType		Public	Property Let
61	DeleteSetting	53	Public	Function
59	InternalReadSetting	60	Private	Function
58	Key	2, 4, 5, 7-9, 11, 14, 18, 21, 29, 33, 45, 46, 48, 49, 52, 53, 57, 59-61	Public	Property Get
58	Key	2, 4, 5, 7-9, 11, 14, 18, 21, 29, 33, 45, 46, 48, 49, 52, 53, 57, 59-61	Public	Property Let
59	MaxStringSize	58, 60	Public	Property Get
58	MaxStringSize	59, 60	Public	Property Let

Pg #	Name	Referenced on Pages	Scope	Type
61	Min	6-17, 22-24, 26-28, 31, 34, 36, 39, 42, 44-47, 50-52, 55, 57, 59, 60, 67, 69, 71-73	Private	Function
59	ReadOnlyReadSetting	4, 8, 48	Public	Function
59	ReadSetting	4, 8, 48, 58, 60	Public	Function
61	RegCreateKey	58, 60	Private	Function
60	WriteSetting	53, 61	Public	Function
62	UCustomButton	63-70		User Control
64	Caption	2-4, 6, 11, 12, 14-17, 20, 22, 41, 47, 56, 65-68	Public	Property Let
64	Caption	2-4, 6, 11, 12, 14-17, 20, 22, 41, 47, 56, 65-68	Public	Property Get
64	CaptionAlignment	65, 68	Public	Property Let
64	CaptionAlignment	65, 68	Public	Property Get
68	DesignTimeOnly	69	Private	Sub
65	DrawButton	68	Private	Sub
65	Enabled	1-3, 5, 6, 9, 11, 12, 14, 15, 47, 64, 67, 68	Public	Property Let
65	Enabled	1-3, 5, 6, 9, 11, 12, 14, 15, 47, 64, 67, 68	Public	Property Get
65	Font	68, 71, 73, 74	Public	Property Set
65	Font	68, 71, 73, 74	Public	Property Get
68	NoPropertySheet		Private	Sub
65	Picture	1, 3, 6, 13, 17, 20, 23, 41-43, 45-47, 63, 64, 66-70	Public	Property Set
65	Picture	1, 3, 6, 13, 17, 20, 23, 41-43, 45-47, 63, 64, 66-70	Public	Property Get
69	PictureFromBitmap	43, 70	Private	Function
69	PictureFromDC	42, 43, 67	Private	Function
69	SetDesignMode	68, 72, 73	Private	Sub
70	TransparentBlit	33, 42, 67	Private	Sub
68	UserControl_AmbientChanged	72	Private	Sub
68	UserControl_InitProperties	72	Private	Sub
65	UserControl_MouseDown		Private	Sub
65	UserControl_MouseMove		Private	Sub
65	UserControl_MouseUp		Private	Sub
68	UserControl_Paint	73	Private	Sub
68	UserControl_ReadProperties	73	Private	Sub
68	UserControl_Show		Private	Sub
68	UserControl_WriteProperties	73	Private	Sub
71	UProgressBar	72-75		User Control
72	BackColor	73, 74	Public	Property Let
72	BackColor	73, 74	Public	Property Get
75	Clear	16, 47, 56, 67, 69, 72, 74	Public	Sub
71	CurrentPercentage	14, 15, 73	Public	Property Get
71	CurrentUnit	73	Public	Property Get
71	EstimatedTimeToFinish		Public	Property Get
72	FillColor	73	Public	Property Let
72	FillColor	73	Public	Property Get
72	FloodStyle	71, 73	Public	Property Let
71	FloodStyle	72, 73	Public	Property Get
71	Font	65, 68, 73, 74	Public	Property Set
71	Font	65, 68, 73, 74	Public	Property Get
73	IncrementProgress	14-16	Public	Sub
72	InvertedTextColor	71, 73, 74	Public	Property Let
72	InvertedTextColor	71, 73, 74	Public	Property Get
72	MaxPercentage	14, 15, 71, 73	Public	Property Let
72	MaxPercentage	14, 15, 71, 73	Public	Property Get
72	MinPercentage	71, 73	Public	Property Let
72	MinPercentage	71, 73	Public	Property Get
71	ProgressUnits	15, 73	Public	Property Let
71	ProgressUnits	15, 73	Public	Property Get
72	SetDesignMode	68, 69, 73	Private	Sub
72	TextAlignment	71, 73, 74	Public	Property Let
72	TextAlignment	71, 73, 74	Public	Property Get
72	UserControl_AmbientChanged	68	Private	Sub
72	UserControl_InitProperties	68	Private	Sub
73	UserControl_Paint	68	Private	Sub
73	UserControl_ReadProperties	68	Private	Sub
73	UserControl_WriteProperties	68	Private	Sub

Table of Contents

Name	Page
frmConfigureMSDE	1
DisplayDialog	1
cmdNav_Click	1
Form_QueryUnload	2
Form_Unload	2
Form_KeyDown	2
tmrConfigureServer_Timer	2
ConfigurationCallback	2
frmGetServerInfo	3
DisplayDialog	3
cmdNav_Click	4
Form_QueryUnload	5
Form_Unload	5
Form_KeyDown	5
ckNTSecurity_Click	5
frmInstallMSDE	6
DisplayDialog	6
cmdNav_Click	6
Form_QueryUnload	7
Form_Unload	7
Form_KeyDown	7
GetAvailableMSIPackageName	7
RemoveMSIFileNameFromArray	9
GetMSIFromProductCode	9
tmrStartInstall_Timer	9
DisableCancelButton	12
ProcessIDFromhWnd	12
frmInstallMSI	13
DisplayDialog	13
cmdNav_Click	13
Form_QueryUnload	13
Form_Unload	13
Form_KeyDown	14
cmdInstallMSI_Click	14
tmrProgress_Timer	14
tmrInstallMSI_Timer	14
InstallMSI	15
frmReadyToInstallMSDE	17
DisplayDialog	17
cmdNav_Click	17
Form_QueryUnload	18
Form_Unload	18
Form_KeyDown	18
frmWelcome	20
DisplayDialog	20
cmdNav_Click	20
Form_QueryUnload	21
Form_Unload	21
Form_KeyDown	21
MGeneralUtilities	30
INDEXTOSTATEIMAGEMASK	30
Max	31
Min	31
MAKELONG	31
MAKELPARAM	31
LOWORD	31
HIWORD	31
MAKEPOINT	31
IsArrayEmpty	31
StripNulls	31
StripLfs	31
Replace	32
Yeild	32
ReturnModuleEXENAME	32
StringComp	32
NumComp	32
CustomFormat	32
TransparentBlit	33
GetOSVersion	34
GetFileVersion	34
CompareFileVersions	35

ReturnPathRoot	35
ReturnFileName	35
ReturnFileExtension	36
Callwcsrchr	36
RemoveFileNameFromPath	36
ReturnShortPathName	36
AddBackslashToPath	37
ReturnComputerName	37
TrueFalseToYesNo	37
FileExists	37
DriveExists	37
CreateTempFileName	37
ReturnProcAddress	38
IsCallExported	38
ReturnFileTime	38
WinTimeToVBATime	38
RemoveIllegalFileChars	39
Is2KShell	39
PointerToStringA	39
PointerToStringW	39
PointerToDWord	40
MHelperFunctions	41
AskAboutCancel	41
CreateTitlePicture	41
PictureFromDC	42
PictureFromBitmap	43
RunningInIde	43
TestIDE	43
MMain	45
TaskbarProxy	45
Main	45
ProcessCommandline	48
ParseServerInfo	49
ChangePassword	49
ChangePasswordNTAuth	49
AddSQLServerLogin	50
AddUserToDatabase	50
AddLoginToServerRoll	50
GetSQLServerVersion	51
IsSAPasswordOK	51
DoesSQLServerExist	51
GetSQLServerRollName	51
IsMSINeeded	52
DBNameIsOK	52
SetRunKey	52
SetCommandLineKey	53
RemoveRunKey	53
RemoveCommandLineKey	53
ConfigureSQLServer	53
Encrypt	56
CRegistry	58
Class_Initialize	58
Key	58
Key	58
DefaultValueType	58
DefaultValueType	58
MaxStringSize	58
MaxStringSize	59
ReadOnlyReadSetting	59
ReadSetting	59
InternalReadSetting	59
WriteSetting	60
DeleteSetting	61
RegCreateKey	61
Min	61
UCustomButton	64
Caption	64
Caption	64
CaptionAlignment	64
CaptionAlignment	64
Enabled	65
Enabled	65
Font	65

Font	65
Picture	65
Picture	65
UserControl_MouseDown	65
UserControl_MouseMove	65
UserControl_MouseUp	65
DrawButton	65
UserControl_AmbientChanged	68
UserControl_InitProperties	68
UserControl_ReadProperties	68
UserControl_Show	68
UserControl_WriteProperties	68
UserControl_Paint	68
NoPropertySheet	68
DesignTimeOnly	68
SetDesignMode	69
PictureFromDC	69
PictureFromBitmap	69
TransparentBlit	70
UProgressBar	71
Font	71
Font	71
EstamatedTimeToFinish	71
CurrentPercentage	71
CurrentUnit	71
ProgressUnits	71
ProgressUnits	71
FloodStyle	71
FloodStyle	72
TextAlignment	72
TextAlignment	72
MaxPercentage	72
MaxPercentage	72
MinPercentage	72
MinPercentage	72
BackColor	72
BackColor	72
FillColor	72
FillColor	72
InvertedTextColor	72
InvertedTextColor	72
SetDesignMode	72
UserControl_AmbientChanged	72
UserControl_InitProperties	72
UserControl_ReadProperties	73
UserControl_WriteProperties	73
UserControl_Paint	73
IncrementProgress	73
Clear	75