

# Re: Communicating between Applications

---

*Source:*

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2006-03/msg01533.html>

---

- *From:* Stefan Berglund <[sorry.no.koolaid@xxxxxx](mailto:sorry.no.koolaid@xxxxxx)>
  - *Date:* Wed, 15 Mar 2006 11:57:08 -0800
- 

On Wed, 15 Mar 2006 11:35:18 -0800, Bob O`Bob <[filterbob@xxxxxxxxxxxxxxxxxxxx](mailto:filterbob@xxxxxxxxxxxxxxxxxxxx)> wrote:  
in <[OONq9eGSGHA.5156@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:OONq9eGSGHA.5156@xxxxxxxxxxxxxxxxxxxxxxxx)>

I've been pondering the appropriateness of mailslots for interprocess comms in a project of my own. Can you point me to any good tutorial/example/sample/demo resources?

Well, there's a demo on Randy Birch's site and Desaware has one as well. I started from there but wasn't satisfied with either approach. For a simple demo that demonstrates that it works, compile this and run it on two machines. Whatever you type on one is shows up on the other. The module that follows is not part of the demo project, but shows a more binary (rather than text) approach that works well for me. The module is used to update the exe across the network and perform regular backups.

' mailslots demo form – basic idea

VERSION 5.00

Begin VB.Form frmMailSlot

Caption = "Mailslot Demo"

ClientHeight = 4290

ClientLeft = 1095

ClientTop = 1515

ClientWidth = 4260

BeginProperty Font

Name = "MS Sans Serif"

Size = 8.25

Charset = 0

Weight = 700

Underline = 0 'False

Italic = 0 'False

Strikethrough = 0 'False

EndProperty

LinkTopic = "Form1"

PaletteMode = 1 'UseZOrder

## Re: Communicating between Applications

```
ScaleHeight = 4290
ScaleWidth = 4260
Begin VB.CommandButton Command1
Caption = "Send"
Height = 375
Left = 180
TabIndex = 2
Top = 3720
Width = 1275
End
Begin VB.TextBox Text1
Height = 1455
Left = 240
MultiLine = -1 'True
TabIndex = 0
Top = 1980
Width = 3795
End
Begin VB.Timer Timer1
Interval = 10
Left = 3960
Top = -60
End
Begin VB.TextBox txtInfo
Height = 1455
Left = 240
MultiLine = -1 'True
TabIndex = 1
Top = 480
Width = 3795
End
End
Attribute VB_Name = "frmMailSlot"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

```
Private Declare Function CloseHandle _
Lib "kernel32" ( _
ByVal hObject As Long) As Long
```

```
Private Declare Function CreateFile _
Lib "kernel32" _
Alias "CreateFileA" ( _
ByVal lpFileName As String, _
ByVal dwDesiredAccess As Long, _
ByVal dwShareMode As Long, _
```

## Re: Communicating between Applications

```
ByVal lpSecurityAttributes As Long, _  
ByVal dwCreationDisposition As Long, _  
ByVal dwFlagsAndAttributes As Long, _  
ByVal hTemplateFile As Long) As Long
```

```
Private Declare Function CreateMailslot _  
Lib "kernel32" _  
Alias "CreateMailslotA" ( _  
ByVal lpName As String, _  
ByVal nMaxMessageSize As Long, _  
ByVal lReadTimeout As Long, _  
ByVal lpSecurityAttributes As Long) As Long
```

```
Const FORMAT_MESSAGE_ALLOCATE_BUFFER = &H100  
Const FORMAT_MESSAGE_FROM_SYSTEM = &H1000  
Const LANG_NEUTRAL = &H0  
Const SUBLANG_DEFAULT = &H1  
Const ERROR_BAD_USERNAME = 2202&
```

```
Private Declare Function FormatMessage _  
Lib "kernel32" _  
Alias "FormatMessageA" ( _  
ByVal dwFlags As Long, _  
lpSource As Any, _  
ByVal dwMessageId As Long, _  
ByVal dwLanguageId As Long, _  
ByVal lpBuffer As String, _  
ByVal nSize As Long, _  
Arguments As Long) As Long
```

```
Private Declare Function GetLastError _  
Lib "kernel32" () As Long
```

```
Private Declare Function ReadFile _  
Lib "kernel32" ( _  
ByVal hFile As Long, _  
lpBuffer As Any, _  
ByVal nNumberOfBytesToRead As Long, _  
lpNumberOfBytesRead As Long, _  
ByVal lpOverlapped As Long) As Long
```

```
Private Declare Function WriteFile _  
Lib "kernel32" ( _  
ByVal hFile As Long, _  
lpBuffer As Any, _  
ByVal nNumberOfBytesToWrite As Long, _
```

## Re: Communicating between Applications

```
lpNumberOfBytesWritten As Long, _  
ByVal lpOverlapped As Long) As Long
```

```
Private Const CREATE_NEW = 1  
Private Const CREATE_ALWAYS = 2  
Private Const OPEN_EXISTING = 3  
Private Const OPEN_ALWAYS = 4  
Private Const TRUNCATE_EXISTING = 5  
Private Const GENERIC_READ = &H80000000  
Private Const GENERIC_WRITE = &H40000000  
Private Const GENERIC_EXECUTE = &H20000000  
Private Const GENERIC_ALL = &H10000000  
Private Const STANDARD_RIGHTS_ALL = &H1F0000  
Private Const INVALID_HANDLE_VALUE = -1  
Private Const FILE_SHARE_READ = &H1  
Private Const FILE_SHARE_WRITE = &H2  
Private Const FILE_ATTRIBUTE_NORMAL = &H80  
Private Const FILE_ATTRIBUTE_HIDDEN = &H2
```

```
Private mSHandle As Long  
Private mCHandle As Long
```

```
Private Sub Command1_Click()
```

```
If (mCHandle <> INVALID_HANDLE_VALUE) Then  
Dim sMsg As String: sMsg = "Release for backup"  
Dim intN As Integer  
For intN = 1 To Len(sMsg)  
Dim BytesWritten As Long  
Dim res As Long: res = WriteFile(mCHandle, CByte(Asc(Mid$(sMsg, intN, 1))), 1, BytesWritten, 0)  
Next  
End If
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
mSHandle = CreateMailslot("\\.\mailslot\horsies", ByVal 0, ByVal 0, ByVal 0)  
If (mSHandle = INVALID_HANDLE_VALUE) Then MsgBox "Unable to open server mailslot"  
Dim sSlotName As String  
sSlotName = "\\*\mailslot\horsies"  
mCHandle = CreateFile(sSlotName, GENERIC_WRITE, FILE_SHARE_READ, 0&, OPEN_EXISTING,  
FILE_ATTRIBUTE_NORMAL, 0&)
```

```
If (mCHandle = INVALID_HANDLE_VALUE) Then  
MsgBox Err.LastDllError  
Dim Buffer As String: Buffer = Space(200)  
FormatMessage FORMAT_MESSAGE_FROM_SYSTEM, ByVal 0&, GetLastError, LANG_NEUTRAL,  
Buffer, 200, ByVal 0&  
MsgBox Buffer
```

## Re: Communicating between Applications

End If

End Sub

Private Sub Form\_Unload(Cancel As Integer)

```
If (mSHandle <> INVALID_HANDLE_VALUE) Then
Call CloseHandle(mSHandle)
mSHandle = 0
End If
```

```
If (mCHandle <> INVALID_HANDLE_VALUE) Then
Call CloseHandle(mCHandle)
mCHandle = 0
End If
```

End Sub

Private Sub Timer1\_Timer()

```
If (mSHandle <> INVALID_HANDLE_VALUE) Then
Dim res As Long: res = 1
Dim BytesRead As Long: BytesRead = 1
Do While (res And BytesRead = 1)
Dim inchar As Byte
res = ReadFile(mSHandle, inchar, 1, BytesRead&, 0)
If (res And BytesRead = 1) Then
With txtInfo
.SelStart = Len(.Text)
.SelText = Chr$(inchar)
End With
End If
Loop
End If
```

End Sub

Private Sub Text1\_KeyPress(KeyAscii As Integer)

```
If (mCHandle <> INVALID_HANDLE_VALUE) Then
Dim res As Long
Dim BytesWritten As Long
res = WriteFile(mCHandle, CByte(KeyAscii), 1, BytesWritten, 0)
KeyAscii = 0
End If
```

End Sub

' module – not related to above demo  
Option Explicit

Re: Communicating between Applications

## Re: Communicating between Applications

### Option Private Module

'--- CreateFile -----

```
Public Const MOVEFILE_REPLACE_EXISTING = &H1
Public Const FILE_ATTRIBUTE_TEMPORARY = &H100
Public Const FILE_BEGIN = 0
Public Const CREATE_NEW = 1
Public Const CREATE_ALWAYS = 2
Public Const OPEN_EXISTING = 3
Public Const OPEN_ALWAYS = 4
Public Const TRUNCATE_EXISTING = 5
Public Const GENERIC_READ = &H80000000
Public Const GENERIC_WRITE = &H40000000
Public Const GENERIC_EXECUTE = &H20000000
Public Const GENERIC_ALL = &H10000000
Public Const INVALID_HANDLE_VALUE = -1
Public Const FILE_SHARE_READ = &H1
Public Const FILE_SHARE_WRITE = &H2
Public Const FILE_ATTRIBUTE_NORMAL = &H80
Public Const FILE_ATTRIBUTE_HIDDEN = &H2
```

```
Public Declare Function CreateFile _
Lib "kernel32" _
Alias "CreateFileA" ( _
ByVal lpFileName As String, _
ByVal dwDesiredAccess As Long, _
ByVal dwShareMode As Long, _
ByVal lpSecurityAttributes As Long, _
ByVal dwCreationDisposition As Long, _
ByVal dwFlagsAndAttributes As Long, _
ByVal hTemplateFile As Long) As Long
```

```
Private Declare Function CreateMailslot _
Lib "kernel32" _
Alias "CreateMailslotA" ( _
ByVal lpName As String, _
ByVal nMaxMessageSize As Long, _
ByVal lReadTimeout As Long, _
ByVal lpSecurityAttributes As Long) As Long
```

```
Private Declare Function GetTickCount _
Lib "kernel32" () As Long
```

```
Public Declare Function KillTimer _
Lib "user32" ( _
ByVal hWnd As Long, _
```

## Re: Communicating between Applications

ByVal nIDEvent As Long) As Long

```
Private Declare Function ReadFile _  
Lib "kernel32" ( _  
ByVal hFile As Long, _  
lpBuffer As Any, _  
ByVal nNumberOfBytesToRead As Long, _  
lpNumberOfBytesRead As Long, _  
ByVal lpOverlapped As Long) As Long
```

```
Public Declare Function SetTimer _  
Lib "user32" ( _  
ByVal hWnd As Long, _  
ByVal nIDEvent As Long, _  
ByVal uElapse As Long, _  
ByVal lpTimerFunc As Long) As Long
```

```
Private Declare Function WriteFile _  
Lib "kernel32" ( _  
ByVal hFile As Long, _  
lpBuffer As Any, _  
ByVal nNumberOfBytesToWrite As Long, _  
lpNumberOfBytesWritten As Long, _  
ByVal lpOverlapped As Long) As Long
```

```
Public Enum ActionCode  
' XcMAC_Compact = 1  
XcMAC_MaxBufLength = 4095  
End Enum
```

```
Public Enum MailSlotCode  
XcMSC_Disconnect = 1  
XcMSC_Reconnect = 2  
XcMSC_Login = 3  
XcMSC_StageVersion = 4  
XcMSC_UpdateVersion = 5  
' XcMSC_RequestCompact = 6  
End Enum
```

```
Public Const mcTimerInterval = 1000  
Private Const mcBackupInterval As Long = 1800000  
Private Const mcKillStagedVersionInterval As Long = 20000
```

```
Private mintAction As Integer  
Private mdblBackupWhen As Double  
Private mlngKillStagedVersionWhen As Long  
Private msStagedVersionPath As String
```

## Re: Communicating between Applications

```
Private mServerHandle As Long
Private mClientHandle As Long
Private mbMailSlots As Boolean
```

```
Public Sub AnnounceLogin()
```

```
If (mbMailSlots) Then PutMailSlot XcMSC_Login, MyComputerName & Chr$(0) & AppVersion
```

```
End Sub
```

```
Public Function Backup(ByRef cn As ADODB.Connection, ByVal sFolder As String) As String
```

```
If (Len(sFolder) > 0) Then
```

```
On Error GoTo BigProblem
```

```
Dim cmd As New ADODB.Command
```

```
With cmd
```

```
.ActiveConnection = cn
```

```
.CommandText = "dbo._Backup"
```

```
.CommandType = adCmdStoredProc
```

```
.NamedParameters = True
```

```
.Parameters.Append .CreateParameter("@BackupFolder", adVarChar, adParamInput, 300, sFolder)
```

```
.Parameters.Append .CreateParameter("@Status", adInteger, adParamOutput, 4)
```

```
.Parameters.Append .CreateParameter("@FileSpec", adVarChar, adParamOutput, 300)
```

```
.CommandTimeout = 0
```

```
.Execute , , adExecuteNoRecords
```

```
Dim lSuccess As Long: lSuccess = .Parameters("@Status")
```

```
Backup = .Parameters("@FileSpec")
```

```
End With
```

```
End If
```

```
If (lSuccess = 0) Then
```

```
SetNextBackupTime
```

```
If (Left(sFolder, 2) <> "\\") Then
```

```
FillFileListBox frmMain.lbxFile, 0, sFolder & "*_ShowTime_Backup"
```

```
With frmMain.lbxFile
```

```
Dim intN As Integer
```

```
For intN = .ListCount - 1 To 0 Step -1
```

```
Dim bMatch As Boolean: bMatch = (InStr(1, .List(intN), "_ShowTime_Backup", vbTextCompare) >= (12 + Len(SQLVersion(cn)) + 2))
```

```
Dim intMax As Integer: If (bMatch) Then intMax = intMax + 1
```

```
Dim strList As String
```

```
If (intMax > 12 Or Not bMatch) Then strList = strList & sFolder & .List(intN) & Chr$(0)
```

```
Next
```

```
End With
```

```
If (Len(Trim$(strList)) > 0) Then
```

```
Dim SHFileOp As SHFILEOPSTRUCT
```

```
With SHFileOp
```

```
.wFunc = FO_DELETE
```

```
.pFrom = strList & Chr$(0)
```

```
.pTo = ""
```

```
.fFlags = FOF_ALLOWUNDO Or FOF_SILENT Or FOF_NOCONFIRMATION
```

```
End With
```

Re: Communicating between Applications

## Re: Communicating between Applications

```
Call SHFileOperation(SHFileOp)
End If
End If
Else
Backup = ""
End If
Exit Function
```

BigProblem:

```
'Debug.Print Err.Description
End Function
'
```

```
'Public Sub CompactNow()
'
```

```
' mlngCompactInterval = mcCompactInterval
' mlngCompactWhen = GetTickCount
'
```

```
'End Sub
```

```
Private Sub FillFileListBox(ByRef lbx As ListBox, ByVal intIndex As Integer, ByVal strPath As String)
```

```
Select Case intIndex
```

```
Dim DDL_FLAGS As Long
```

```
Case 0: DDL_FLAGS = DDL_EXCLUSIVE Or DDL_ARCHIVE
```

```
Case 1: DDL_FLAGS = DDL_EXCLUSIVE Or DDL_ARCHIVE Or DDL_DIRECTORY
```

```
Case 2: DDL_FLAGS = DDL_EXCLUSIVE Or DDL_HIDDEN
```

```
Case 3: DDL_FLAGS = DDL_EXCLUSIVE Or DDL_SYSTEM
```

```
Case 4: DDL_FLAGS = DDL_EXCLUSIVE Or DDL_DIRECTORY
```

```
Case 5: DDL_FLAGS = DDL_EXCLUSIVE Or DDL_DRIVES
```

```
Case Else
```

```
End Select
```

```
' Dim WIN As WinVersion
```

```
' GetWinVersion WIN
```

```
' If (WIN.IsW2K Or WIN.IsWXP) Then
```

```
' Call SendMessage(lbx.hWnd, LB_RESETCONTENT, 0&, ByVal 0&)
```

```
' Call SendMessage(lbx.hWnd, LB_DIR, DDL_FLAGS, ByVal strPath & vbNullChar)
```

```
' With lbx
```

```
' Dim intN As Integer
```

```
' For intN = .ListCount - 1 To 0 Step -1
```

```
' .List(intN) = Mid$(.List(intN), 1, Len(.List(intN)))
```

```
' Next
```

```
' End With
```

```
' Else
```

```
' frmMain.Dir1.Path = strPath
```

```
' With frmMain.File1
```

```
' MsgBox strPath & "=" & CStr(.ListCount)
```

```
' For intN = 0 To .ListCount - 1
```

```
' lbx.AddItem Left$(.List(intN), Len(.List(intN)))
```

```
' Next
```

```
' End With
```

Re: Communicating between Applications

## Re: Communicating between Applications

```
' End If
```

```
Call SendMessage(lbx.hWnd, LB_RESETCONTENT, 0&, ByVal 0&)
```

```
Call SendMessage(lbx.hWnd, LB_DIR, DDL_FLAGS, ByVal strPath & vbNullChar)
```

```
With lbx
```

```
Dim intN As Integer
```

```
For intN = .ListCount - 1 To 0 Step -1
```

```
.List(intN) = Mid$(.List(intN), 1, Len(.List(intN)))
```

```
Next
```

```
End With
```

```
End Sub
```

```
Private Function GetMailSlot(ByVal lngLength As Long) As String
```

```
Dim bytBuf() As Byte: ReDim bytBuf(0 To lngLength - 1)
```

```
GetMailSlot = Space(lngLength)
```

```
Dim lngBytesRead As Long
```

```
Dim lngResult As Long: lngResult = ReadFile(mServerHandle, bytBuf(0), lngLength, lngBytesRead, 0&)
```

```
If (lngBytesRead = lngLength And lngResult > 0) Then GetMailSlot = StrConv(bytBuf, vbUnicode)
```

```
End Function
```

```
Public Sub InitMailSlot()
```

```
mServerHandle = CreateMailslot("\\.\mailslot\SQLhorsies", 0&, 0&, 0&)
```

```
mClientHandle = CreateFile("\\*\mailslot\SQLhorsies", GENERIC_WRITE, FILE_SHARE_READ, 0&, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0&)
```

```
mbMailSlots = ((mServerHandle <> INVALID_HANDLE_VALUE) And (mClientHandle <> INVALID_HANDLE_VALUE))
```

```
End Sub
```

```
Public Sub KillMailSlot()
```

```
If (mServerHandle <> INVALID_HANDLE_VALUE) Then
```

```
Call CloseHandle(mServerHandle)
```

```
mServerHandle = 0
```

```
End If
```

```
If (mClientHandle <> INVALID_HANDLE_VALUE) Then
```

```
Call CloseHandle(mClientHandle)
```

```
mClientHandle = 0
```

```
End If
```

```
End Sub
```

```
Public Sub PutMailSlot(ByVal lngMailSlotCode As Long, Optional ByVal strBuf As String)
```

```
If (mbMailSlots And (mClientHandle <> INVALID_HANDLE_VALUE)) Then
```

```
Dim lngBuf As Long: lngBuf = &HFCF00000 Or (lngMailSlotCode * (XcMAC_MaxBufLength + 1)) Or (Len(strBuf) And XcMAC_MaxBufLength)
```

Re: Communicating between Applications

## Re: Communicating between Applications

```
Dim lngBytesWritten As Long
Dim lngResult As Long: lngResult = WriteFile(mClientHandle, lngBuf, 4&, lngBytesWritten, 0&)
If (lngBytesWritten = 4) Then
If (Len(strBuf) > 0) Then
Dim bytBuf() As Byte: bytBuf = StrConv(strBuf, vbFromUnicode)
lngResult = WriteFile(mClientHandle, bytBuf(0), Len(strBuf), lngBytesWritten, 0&)
End If
End If
End If

End Sub

Public Sub SetNextBackupTime()

mdblBackupWhen = CDbl(GetTickCount) + CDbl(mcBackupInterval)

End Sub

Public Sub TimerCallback(ByVal hWnd As Long, ByVal uMsg As Long, _
ByVal idEvent As Long, ByVal dwTime As Long)

With frmMain
Dim lngResult As Long, lngInBuf As Long, lngBytesRead As Long
lngResult = ReadFile(mServerHandle, lngInBuf, 4, lngBytesRead, 0)
If (lngResult And lngBytesRead = 4) Then
Dim frm As Form
For Each frm In Forms
Dim bDialog As Boolean
If (InStr(1, frm.Name, "dlg") = 1) Then bDialog = True
Next
Dim lngCommand As Long: lngCommand = lngInBuf And (&HFFF0000 Or XcMAC_MaxBufLength *
(XcMAC_MaxBufLength + 1))
Dim lngLength As Long: lngLength = lngInBuf And XcMAC_MaxBufLength

Select Case lngCommand
Case CLng(&HFCF00000 Or (XcMSC_Disconnect * (XcMAC_MaxBufLength + 1)))
If (lngLength > 0) Then
Dim sTarget As String: sTarget = GetMailSlot(lngLength)
If (Not bDialog) Then
dlgMessage.SetMessage "Disconnecting..."
.MenuBusy = True
End If
End If

Case CLng(&HFCF00000 Or (XcMSC_Reconnect * (XcMAC_MaxBufLength + 1)))
If (.MenuBusy) Then
mintAction = 0
ClearMessage
.MenuBusy = False
If (.CurrentForm) <> XtabReport Then SelectMenu .CurrentForm
End If
```

## Re: Communicating between Applications

```
Case CLng(&HF0000 Or (XcMSC_Login * (XcMAC_MaxBufLength + 1)))
If (lngLength > 0) Then
Dim sLoginName As String: sLoginName = GetMailSlot(lngLength)
Dim intN As Integer: intN = InStr(1, sLoginName, Chr$(0))
Dim sLoginVersion As String: sLoginVersion = Mid$(sLoginName, intN + 1)
If (intN > 0) Then sLoginName = Mid$(sLoginName, 1, intN - 1)
End If
If (IsServer(frmMain.Server)) Then
' Dim rst As ADODB.Recordset
' Set rst = .Conn.Execute("SELECT Computer,Version FROM Logins ORDER BY Version DESC")
' With rst
' If (Not .EOF) Then
' Dim sComputerName As String: sComputerName = .Fields("Computer").Value
' Dim sVersion As String: sVersion = .Fields("Version").Value
' If (Len(sLoginVersion) > 0) Then
' If (sLoginVersion > sVersion) Then
' sComputerName = sLoginName
' Dim bUpdate As Boolean: bUpdate = True
' End If
' End If
' .MoveNext
' Do While (Not .EOF)
' If (.Fields("Version").Value <> sVersion) Then bUpdate = True
' .MoveNext
' Loop
' .Close
' If (bUpdate Or ((sLoginName <> sComputerName) And (sLoginVersion <> sVersion))) Then
' PutMailSlot XcMSC_StageVersion, sComputerName
' End If
' End If
' End With
Const cComputer = 0
Const cVersion = 1
Dim ar: ar = GetArray(.Conn, "SELECT Computer,Version FROM Logins ORDER BY Version DESC")
If (IsArray(ar)) Then
Dim sComputerName As String: sComputerName = ar(cComputer, 0)
Dim sVersion As String: sVersion = ar(cVersion, 0)
If (Len(sLoginVersion) > 0) Then
If (sLoginVersion > sVersion) Then
sComputerName = sLoginName
Dim bUpdate As Boolean: bUpdate = True
End If
End If
Dim Row As Long
For Row = 1 To UBound(ar, 2)
If (ar(cVersion, Row) <> sVersion) Then bUpdate = True
Next
If (bUpdate Or ((sLoginName <> sComputerName) And (sLoginVersion <> sVersion))) Then
PutMailSlot XcMSC_StageVersion, sComputerName
End If
```

## Re: Communicating between Applications

```
End If  
End If
```

```
Case &HFCF00000 Or (XcMSC_StageVersion * (XcMAC_MaxBufLength + 1))  
If (lngLength > 0) Then  
If (GetMailSlot(lngLength) = MyComputerName) Then  
msStagedVersionPath = GetUpdateFolder(.Conn)  
If ((Len(msStagedVersionPath) > 0) And (Left(msStagedVersionPath, 2) = "\\")) Then  
CopyFile AppPath(App.Path) & "SQLShowTimeLoader.exe", msStagedVersionPath &  
"SQLShowTimeLoader.exe", 0  
CopyFile AppPath(App.Path) & "SQLShowTime.exe", msStagedVersionPath & "SQLShowTime.exe", 0  
PutMailSlot XcMSC_UpdateVersion, AppVersion  
mlngKillStagedVersionWhen = dwTime + mcKillStagedVersionInterval  
End If  
End If  
End If
```

```
Case &HFCF00000 Or (XcMSC_UpdateVersion * (XcMAC_MaxBufLength + 1))  
If (lngLength > 0) Then  
sVersion = GetMailSlot(lngLength)  
If (sVersion > AppVersion) Then  
If (Not bDialog) Then  
dlgMessage.SetMessage "Auto Updating..."  
Sleep 400  
End If  
msStagedVersionPath = GetUpdateFolder(.Conn)  
If (Left(msStagedVersionPath, 2) = "\\") Then  
Dim sAppPath As String: sAppPath = AppPath(App.Path)  
Dim sApp As String: sApp = sAppPath & "SQLShowTimeLoader."  
KillFile sApp & "old": CopyFile sApp & "exe", sApp & "old", 0  
CopyFile msStagedVersionPath & "SQLShowTimeLoader.exe", sApp & "exe", 0  
sApp = sAppPath & "SQLShowTime."  
KillFile sApp & "old": CopyFile sApp & "exe", sApp & "old", 0  
CopyFile msStagedVersionPath & "SQLShowTime.exe", AppPath(GetTempDir()) & "SQLShowTime.exe", 0  
ShellExecute 0&, "Open", AppPath(App.Path) & "SQLShowTimeLoader.exe", "/v " & .Server, App.Path,  
SW_SHOWNORMAL  
UnloadAll  
Exit Sub  
Else  
ClearMessage  
End If  
End If  
End If
```

```
''' Case &HFCF00000 Or (XcMSC_RequestCompact * (XcMAC_MaxBufLength + 1))  
''' If (bMaster) Then CompactNow  
End Select  
End If
```

```
' If (mintAction > 0) Then  
' Select Case mintAction
```

## Re: Communicating between Applications

```
"" Case XcMAC_Compact
"" Dim xConn As ADODB.Connection
"" Set xConn = New ADODB.Connection
"" On Error Resume Next
"" xConn.Open mcConnectionString & sPath & ";Mode=Share Exclusive;"
"" If (Err.Number <> 0) Then
"" Err.Clear
"""" If (dwTime > (mInngCompactWhen + mcCompactTimeout)) Then
"""" PutMailSlot XcMSC_Reconnect
"""" mInngCompactWhen = dwTime + mcCompactReset
"""" End If
"" Else
"" If (xConn.State And adStateOpen <> 0) Then xConn.Close
"""" If (CompactDB(sPath, sPath)) Then
"""" PutMailSlot XcMSC_Reconnect
"""" mInngCompactWhen = dwTime + mInngCompactInterval
"""" End If
"" End If
"" On Error GoTo 0
' End Select
' End If

If (IsServer(frmMain.Server) And dwTime > mdbBackupWhen) Then
dlgMessage.SetMessage "Performing Backup...", "Please Wait"
Backup .Conn, AppPath(.BackupFolder)
mdbBackupWhen = dwTime + mcBackupInterval
ClearMessage
End If

"" If (.IsServer) Then
"" If (Not .MenuBusy) Then
"" If ((dwTime > mInngCompactWhen) And (mInngCompactInterval <> 0)) Then
"" PutMailSlot XcMSC_Disconnect, sPath
"" mintAction = XcMAC_Compact
"" End If
"" End If
"" End If

If (mInngKillStagedVersionWhen > 0) Then
If (dwTime > mInngKillStagedVersionWhen) Then
KillFile msStagedVersionPath
mInngKillStagedVersionWhen = 0
End If
End If
End With

End Sub
```

---

This posting is provided "AS IS" with no warranties, no guarantees, and no conferred rights.

Stefan Berglund

.