Mise à jour automatique d'un frontal.

Ce que tout le monde veut pour les mises à jour de frontal access c'est d'avoir le minimum de choses à gérer.
Je suis sur cette affaire depuis un moment déjà, et le hasard de la navigation dans les anneaux access m'a fait tomber sur un script incroyable. Ce script à l'origine était destiné au compactage à la demande avec relance du frontal.
J'ai sauté dessus et je l'ai adapté pour les mises à jour de frontaux. Pour l'utiliser il faut juste avoir une ligne qui retourne aux conditions de licence chez « creative commons ».

Dans le frontal une table locale « version » avec :

NumVersion (texte) : Le numéro de version pour le « versionning » du développeur
ladate (date) : La date de mise à jour On utilise la date pour effectuer le test.
NomCompletFrontalMaJ (texte) : L'emplacement et le nom du nouveau frontal, (il peut être différent de celui de l'utilisateur) ex la mise à jour est FRT5200.accdr et l'utilisateur fait tourner FRTJEANJACQUES.accdr.

On test la présence du fichier à l'emplacement enregistré dans la table
On test les dates des 2 frontaux local et distant
Si c'est supérieur alors on envoi la mise à jour.

Pour utiliser le code c'est :

```
If VerificationMaJ <> "Aucun" Then
   MsgBox "Une mise à jour à faire !"
   Restart
End If
```

```
' Licence to http://creativecommons.org/licenses/by/3.0/
Option Compare Database
Option Explicit

Private Const TIMEOUT = 60

Public Sub Restart(Optional Compact As Boolean = False)
   ' Compact pour un compactage !!
   Dim scriptpath As String

   ' Construct the full path of our temporary batch file
   scriptpath = Application.CurrentProject.FullName & ".dbrestart.bat"

   ' if the script already exists, then check it isn't an old remnant
   ' that has passed it's timeout.
   If Dir(scriptpath, vbNormal) <> "" Then
      If DateAdd("s", TIMEOUT * 2, FileDateTime(scriptpath)) < Date Then
         ' We've passed twice the batch file timeout, giving ample time for
         ' it to exectute, so if it's still there, it's most probably a dud

         Kill scriptpath
      Else
         ' Timeout hasn't expired beyond the acceptable limit, so it's probably
         ' still active, just try to close the application again
         Application.Quit acQuitSaveAll
         Exit Sub
      End If
   End If
```

```
' Construct the batch file
' Note that the TIMEOUT value is only used as a loop counter and
' we do not really count elapsed time in the batch file.
' The ping command takes some time to load and start and even though
' we set its timeout to 100ms, it will take much longer than that to
' execute.
' If we've been asked to comnpact the database, we launch the database
' using the /compact command line switch

Dim s As String
s = s & "SETLOCAL ENABLEDELAYEDEXPANSION" & vbCrLf
s = s & "SET /a counter=0" & vbCrLf
s = s & ":CHECKLOCKFILE" & vbCrLf
s = s & "ping 0.0.0.255 -n 1 -w 100 > nul" & vbCrLf
s = s & "SET /a counter+=1" & vbCrLf
s = s & "IF ""!counter!""==""" & TIMEOUT & """ GOTO CLEANUP" & vbCrLf
s = s & "IF EXIST ""%~f2.%4"" GOTO CHECKLOCKFILE" & vbCrLf

If VerificationMaJ <> "Aucun" Then
   s = s & "COPY " & VerificationMaJ & " " & Application.CurrentProject.FullName & vbCrLf
End If

If Compact Then
   s = s & """%~f1"" ""%~f2.%3"" /compact" & vbCrLf
End If

s = s & "start "" "" ""%~f2.%3""" & vbCrLf
s = s & ":CLEANUP" & vbCrLf
s = s & "del %0"

' Write batch file
Dim intFile As Integer
intFile = FreeFile()
Open scriptpath For Output As #intFile
Print #intFile, s
Close #intFile


' Create the arguments to be passed to the script
' Here we pass it the full path to the database minus the extension which we pass separately
' this is done so that we can reconstruct the path to the lock file easily in the script.
' The extension to the lock file is also passed as a third argument.
Dim dbname As String, ext As String, lockext As String, accesspath As String
Dim idx As Integer

' Get the path to the msaccess executable, wherever that is
accesspath = SysCmd(acSysCmdAccessDir) & "msaccess.exe"

' Find the extension, starting from the end
For idx = Len(CurrentProject.FullName) To 1 Step -1
   If Mid(CurrentProject.FullName, idx, 1) = "." Then Exit For
Next idx
dbname = left(CurrentProject.FullName, idx - 1)
ext = Mid(CurrentProject.FullName, idx + 1)

' Depending on the database extension, determine its lock file extension
If left(ext, 2) = "ac" Then
```

```
      lockext = "laccdb"
   Else
      lockext = "ldb"
   End If

   ' Call the batch file
   s = """" & scriptpath & """ """ & accesspath & """ """ & dbname & """ " & ext & " " & lockext
    Shell s, vbHide

   ' Close our application
  Application.Quit acQuitSaveAll
End Sub

Function VerificationMaJ() As String
'Fonction de test général
Dim EmplacementFrt As String
Dim ladateLocal
Dim ladateFrontalMaJ

Dim rst As New ADODB.Recordset

rst.Open "Version", CurrentProject.Connection, adOpenStatic, adLockReadOnly
rst.MoveFirst
ladateLocal = rst!ladate
EmplacementFrt = Nz(rst!NomCompletFrontalMaJ)
rst.Close: Set rst = Nothing

If FichierExiste(EmplacementFrt) = True Then
   'Recherche la date sur Frontal MàJ
   ladateFrontalMaJ = donneladate(EmplacementFrt)
   Else
   'Le fichier de MàJ n'existe pas
   VerificationMaJ = "Aucun"
   Exit Function
End If


If ladateFrontalMaJ > ladateLocal Then
   VerificationMaJ = EmplacementFrt
   Else
   VerificationMaJ = "Aucun"
End If


End Function
Function donneladate(kelbase)
'Connexion au frontal et interrogation
   Dim conn, strConnect, rs
   Dim leSQL As String
   Set conn = CreateObject("ADODB.Connection")
   strConnect = "Provider=Microsoft.Ace.OLEDB.12.0;Data Source=" & kelbase & ";Persist Security Info=False;"
   conn.Open strConnect
   leSQL = "SELECT Version.ladate FROM Version"
   Set rs = conn.Execute(leSQL)
   rs.MoveFirst
   donneladate = rs(0)
   conn.Close
```

```
   Set conn = Nothing
End Function
Function FichierExiste(leFichier As String) As Boolean
   Dim filesys
   Set filesys = CreateObject("Scripting.FileSystemObject")
   If filesys.FileExists(leFichier) Then
      FichierExiste = True
   End If
End Function
```