



Login

Register

[MS SQL](#)
[Oracle](#)
[DB2](#)
[Access](#)
[MySQL](#)
[PostgreSQL](#)
[Sybase](#)
[PHP](#)
[SQL Etc](#)
[SQL Scripts & Samples](#)
[Links](#)
[Database Forum](#)
[Database Journal](#) | [DBA Support](#) | [SQLCourse](#) | [SQLCourse2](#)
[» Database Journal Home](#)
[» Database Articles](#)
[» Database Tutorials](#)
[MS SQL](#)
[Oracle](#)
[DB2](#)
[MS Access](#)
[MySQL](#)
[» RESOURCES](#)
[Database Tools](#)
[SQL Scripts & Samples](#)
[Links](#)
[» Database Forum](#)
[» Slideshows](#)
[» Sitemap](#)
**Free Newsletters:**
☒ [DatabaseDaily](#)
**News Via RSS Feed**


## Featured Database Articles

### [MS Access](#)

Posted Dec 19, 2003

## Automatically Deploy a New Access Client

 By [Danny Lesandrini](#)

How do you manage the distribution of new versions of your Microsoft Access applications to network users? I am sure there are third-party tools you can buy, but it seems like such a waste. All you want to do is:

- figure out if a newer version is available
- give the user the option to get latest
- copy down the new mdb file from the server
- reopen the application

Sounds simple, I know, but if it hasn't already struck you, there is a little problem with this process: a new version of the client file cannot be copied over top of the old one while it is in use. The work-around for this problem requires us to jump around between files, but to the end user the update will appear seamless. Follow along as we walk through the process and be sure to [download](#) the code associated with this article.

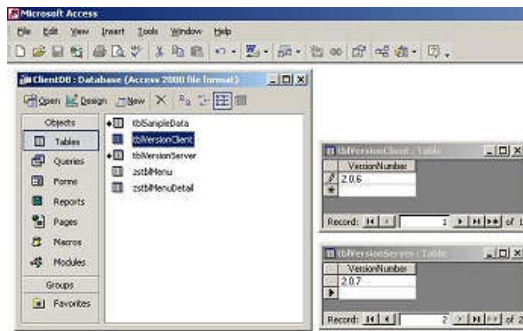
### Getting started

Before we get into the code, we need to take care of some housekeeping. First, this solution assumes that you have split your data out from the client and it is stored in a central location on a network server somewhere. This common network location is very important to the success of this solution, in as much as its a repository for the following:

1. A master table with the most current version number.
2. A master copy of the latest version of your mdb file.

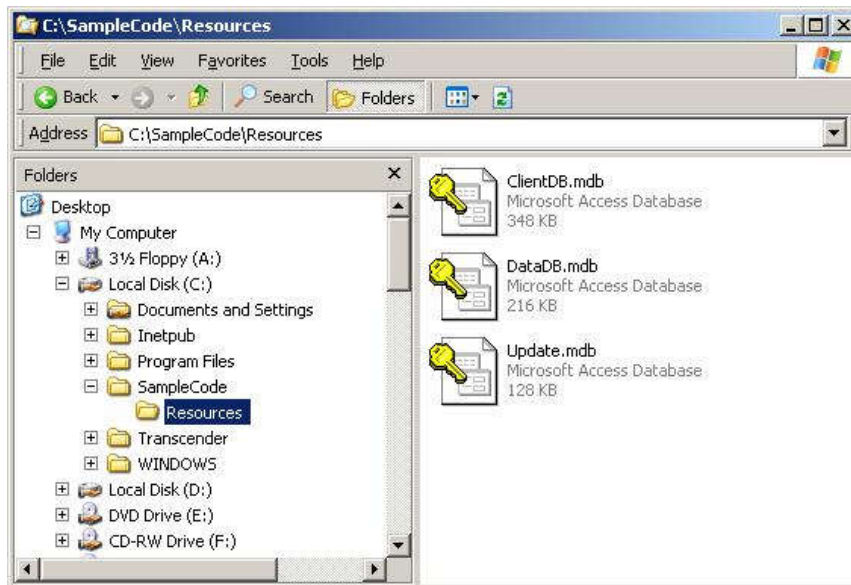
So, let's begin by creating the tables to hold version numbers. You will need one table in the data file on the server, named appropriately enough, tblVersionServer and another in the client named tblVersionClient. The client will link to the version table in the server so both are available for comparison, as shown in the image below.

[Click for larger image](#)



Notice in the example that both tables contain only one column and only one record. In this example, (taken from the [download code](#) associated with the article), the client is version 2.0.6 while the server indicates that version 2.0.7 is available. When the client application opens, the first order of business is to compare these two versions to see if it happens to be the latest. If not, then the process for downloading the latest client is initiated.

It should be noted that the names and locations of all the files involved are rather important. Consider the file example below. (While this is on my C:\ drive, it would normally be placed on the network in a file share accessible to all.)



For the sake of this article, I named the common network share "Resources." The sample code specifically looks for a folder with this name. It also expects to see a data file by the name DataDB.mdb, a client file by the name ClientDB.mdb and a utility database with the name Update.mdb.

If you unpack the code for this article, the client will automatically relink to the data, so long as it finds the DataDB.mdb file in a subfolder named "Resources." While I do not like the idea of hard coding file names and paths, it can be generally assumed that the location of the data file will remain constant, so this should not create any kind of hardship. To put this into production, you will need to modify the code to match your file names and paths.

## The Smart Client

As mentioned above, the client application contains both version tables: tblVersionClient as a local table and tblVersionServer as a linked table. Accordingly, it is easy to perform a comparison of the version values in these two tables when your application opens. I use a form named frmSplash, which has the company logo and copyright notice as a start-up form. This form checks table links and version number. The code looks something like this:

```
Option Compare Database
Option Explicit
```

```

Private strVerClient As String
Private strVerServer As String

Private Sub Form_Load()
On Error Resume Next

' Populate module level variables when form loads.
strVerClient = Nz(DLookup("[VersionNumber]", "[tblVersionClient]"), "")
strVerServer = Nz(DLookup("[VersionNumber]", "[tblVersionServer]"), "")

End Sub

Private Sub Form_Timer()
On Error Resume Next

Dim strMsg As String
Dim strPath As String
Dim strUpdateTool As String
Const q As String * 1 = ""

Me.TimerInterval = 0

' If versions match, then proceed with opening of main form.
If strVerClient = strVerServer Then
Me.Visible = False
DoCmd.OpenForm "fmnuMain"

' ... if not, then offer the user the option to download latest.
Else
strMsg = "You do not have the correct version." & vbCrLf & vbCrLf & _
"Would you like to download the latest client?"
If MsgBox(strMsg, vbExclamation + vbOKCancel, "Update") = vbOK Then

' Notice that I'm using a custom function, LastInStr(), to find
' the last instance of a character. Newer versions of Access expose
' the InstrRev() function to accomplish this, but Access 97 does not.
strPath = Left(CurrentDb.Name, LastInStr(CurrentDb.Name, "\"))
strPath = strPath & "Resources\Update.mdb"
' Enclose the file path in quotes to avoid problems with spaces
' in file and/or folder names.
strUpdateTool = "MSAccess.exe " & q & strPath & q

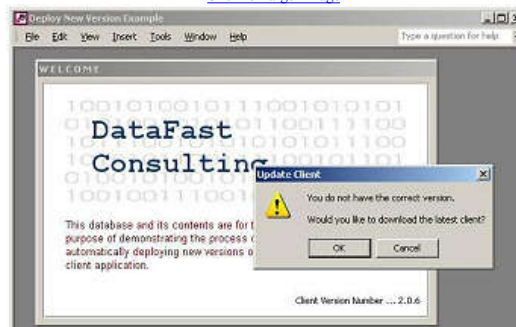
' This is where the real work happens.
' Use SHELL command to open the UPDATE.MDB utility
' ... then quit this client so it may be overwritten.
Shell strUpdateTool, vbNormalFocus
DoCmd.Quit

End If
End If
End Sub

```

When a version mismatch is found, the user is informed and given the opportunity to download the latest copy of the client. (Notice in the image below that the splash screen displays the client version as 2.0.6 and we know from the discussion above that the server is showing that 2.0.7 is available.) If the user clicks OK then the ClientDB.mdb file is closed and the Update.mdb file is opened, but in most cases, this appears to happen within the current session of Microsoft Access.

[Click for larger image](#)



## Performing the Update

Once the Update.mdb utility opens, it goes right to work. Once again, the form open event is used to capture file and names and paths into module level variables. I have also chosen to use this event to make a backup copy of the user's client ... just in case there is a problem with the new version. After making a backup, I delete the old client file, and we are ready for the next step.

In order to allow time for the backup to be created, I place the file copy code in the form timer event. This little pause is sufficient to ensure that the backup will have been successfully created and the original client file removed. From here, it is a simple file copy command and then we use the SHELL command to reopen our newly copied client. A final DoCmd.Quit will close the update utility and the user is now presented with the latest client. All of this is accomplished in the few lines of code below.




---

```
Option Compare Database
Option Explicit
```

```
Dim strPath As String
Dim strDest As String
Dim strBkup As String
Dim strMyDB As String
Dim strVer As String
```

---

```
Private Sub Form_Open(Cancel As Integer)
On Error Resume Next

' Update status form to identify version being copied.
strVer = DLookup("[VersionNumber]", "tblVersionServer")
Me.txtVer.Caption = "Installing version number ... " & strVer

' Load variables with correct file name-path values.
strMyDB = CurrentDb.Name
strPath = Left(strMyDB, LastInStr(strMyDB, "\"))
strDest = Replace(strPath, "Resources\", "ClientDB.mdb")
strBkup = Replace(strPath, "Resources\", "ClientDB_bkup.mdb")

' Stop processing here so article readers may step through code.
Stop

' Create a backup (replacing existing backup if necessary) and
' remove the target file.
If Dir(strBkup) <> "" Then Kill strBkup
FileCopy strDest, strBkup
If Dir(strDest) <> "" Then Kill strDest

End Sub
```

---

```
Private Sub Form_Timer()
On Error Resume Next

Dim strSource As String
Dim strMsg As String
Dim strOpenClient As String
Const q As String = ""

' We make the assumption that the new client is in the
```

```
' same folder as this utility.
strSource = strPath & "ClientDB.mdb"
FileCopy strSource, strDest

' Now that the new client file has been copied, it may
' be opened. Use the SHELL command to accomplish this.
strOpenClient = "MSAccess.exe " & q & strDest & q
Shell strOpenClient, vbNormalFocus

' Exit from this application.
DoCmd.Quit
```

End Sub

## Final thoughts

If you [download](#) the code for this article, you will be able to run through the process and observe the steps more closely. I have added a button to the main form for resetting the client version back to 2.0.6 and another to force the update regardless of the version. The code includes a cool function for relinking tables, which is always useful to have laying around, and that alternate to the InstrRev() function I use called LastInStr(). (This function, LastInStr() is not my own code, but has been around forever. My thanks to its author, whoever that may be.)

I recently got involved in a project where they approach this problem from a different slant. The developer created a batch file that copies the latest client from the server to a local folder on the user's machine. The desktop shortcut the user clicks to launch the program opens NOT the Access mdb file, but the batch file. Each time the user clicks the link, a new version is copied down from the server.

```
REM          This batch file copies the XYZ client from  \\xyz001\  server
REM          ... and starts the program.

COPY \\xyz001\XYZClient.mdb c:\XYZ\XYZClient.mdb
START "MSAccess.exe" c:\XYZ\XYZClient.mdb

EXIT
```

While this method is much simpler than the one described above, it is a hog for network bandwidth and should be considered only where the client is small and there are few users. I have included it here to show that I am aware there is more than one approach to this issue. No doubt, many readers have handled this in different ways and as usual, I look forward to your feedback.

» [See All Articles by Columnist Danny J. Lesandrini](#)

## [MS Access Archives](#)

### 5 Comments ([click to add your comment](#))

By Kurt September 19 2014 11:02 PDT

Danny, Thank you for posting this article. I had already written a similar application but it was not working well due to the amount of security on the servers this is being used on. Reading through this encouraged me to re-visit my app and see if there was code you had provided that could be of assistance. My application runs a little differently. I have a configuration app for storing the locations and configurations of multiple applications to be updated. I use custom properties instead of tables for version control. This last piece allows the use of the same Client with different DataDB files. (multiple locations with different data) Thank you again! K

[Reply to this comment](#) By Edward June 27 2014 19:36 PDT



A good solution indeed - but i just want to point out, the function used to find the path of the database "CurrentDb.Name" - in any scenario - would only give you the path of the front end part of a split ms access database. There is no mention of a function that can access the back-end ms access database file. Thanks once again for this solution using Access rather than any third party tools.

[Reply to this comment](#) By No One Important July 14 2009 10:43 PDT



I can NOT wait to try this.

[Reply to this comment](#) By Jennifer Reusser April 29 2009 15:58 PDT



I am currently using a method similar to the shortcut pointing to the batch file. However, I also have my batch file check to see if the current version of the front-end is already on the local machine using a filename including the version number and removed any old versions first. I also need to limit the check to just the current project (we have multiple projects doing the same thing)... REM This batch file copies the most up-to-date Project interface from the server to your local machine REM ... and starts the program if not exist C:\XYZ mkdir C:\XYZ if not exist c:\XYZ\ProjectInterface###.mdb if exist C:\XYZ\ProjectInterface\*.mdb del C:\XYZ\ProjectInterface\*.mdb if not exist c:\XYZ\ProjectInterface###.mdb xcopy Interface\ProjectInterface###.mdb c:\XYZ /q /y start msaccess.exe c:\XYZ\ProjectInterface###.mdb This way the front-end is only copied from the network if it has been updated since the last time the user access the file.

[Reply to this comment](#) By Kingkef February 25 2014 07:08 PST



@Jennifer can you provide a sample copy of that batch file so I can try your method. I am a noob so and have been running around like headless chicken updating these computer to the latest front end. also thanks to Danny for an amazing 'how to' article.



Comment and Contribute

Your name/nickname

Your email

Subject

(Maximum characters: 1200). You have **1200** characters left.



[Privacy & Terms](#)



Submit Your Comment

Latest Forum Threads

MS Access Forum

Topic

By

Replies

Updated

<a href="#">Help With Microsoft Access</a>	<a href="#">kasy</a>	<a href="#">0</a>	<a href="#">September 4th, 07:35 PM</a>
<a href="#">Linked table not sorting or filtering - ODBC error</a>	<a href="#">Java</a>	<a href="#">1</a>	<a href="#">August 28th, 10:37 AM</a>
<a href="#">Use Parameter in select statement (Sql in Microsoft Access)</a>	<a href="#">katty.jonh</a>	<a href="#">1</a>	<a href="#">July 25th, 06:45 AM</a>
<a href="#">Query Issue</a>	<a href="#">algebroni</a>	<a href="#">7</a>	<a href="#">July 23rd, 04:22 PM</a>



Property of Quinstreet Enterprise.

[Terms of Service](#) | [Licensing & Reprints](#) | [About Us](#) | [Privacy Policy](#) | [Advertise](#)  
Copyright 2016 QuinStreet Inc. All Rights Reserved.