# Solving the Access Driver Problem on ColdFusion 8 - 64 bit

Posted At : April 7, 2009 1:57 PM | Posted By : Mark Kruger
Related Categories: ColdFusion

First off, let me say that I love ColdFusion 8 on the 64 bit JVM. If you have a data driven site that requires extra processing power you should definitely move in that direction. However, there is one thing that really causes some consternation about the platform - the lack of Jet drivers for MS Access. I know I know, I have spoken with open disdain about "using Access in production" and in the words of Dan Quayle, I stand by all my misstatements. Still, there *is one thing* that I do use Access for in production - as a portable export utility.

Using the proxy technique I describe in my post counterintuitively titled **Using a DSN Connection for Connectionless Access** I have several applications that crunch numbers or develop report data, then file them away in tables in an MDB File which is zipped up for download or emailing. It works well and Access is so much more useful than Excel for some savvy corporate analysts. Now along comes 64 bit Windows and suddenly there are no (zip, nada, zilch) Jet drivers for connecting from Coldfusion.

Luckily, I have my wits about me. Also luckily I have a 32 bit server at my disposal as well (it means "available to use" - I'm not implying there is a server near my kitchen sink). So I came up with this clever solution.

### Set up Access

I configured the 32 bit CF server to host the Access DB and I set up the DSN. For example, let's say I added an Access DSN to the 32 bit server and named it "AccessProxy".

### Install Remote Web Service CFC

Next, I installed a CFC to act as a proxy. It's a pretty simple piece of code.

```
<cfcomponent displayname="Access Passthrough Component">

<cffunction name="runQ" returntype="query" access="remote">

    <cfargument name="dsn" type="string" required="Yes"/>
    <cfargument name="qString" required="yes">

    <Cfset var qry = ''/>

    <cfquery name="qry" datasource="#dsn#">
        #preservesinglequotes(qString)#
    </cfquery>


    <cfreturn qry/>

</cffunction>

</cfcomponent>
```

I also added code to restrict the running of this function to specific IP address calls - that way it could only be called from one of my internal servers.

### Add a Web Service With a Duplicate DSN Name

This is where I get really clever. Remember, I'm trying to get some existing code to work on the 64 bit server and I want to avoid wholesale code changes. I do *not* want to replace all that Cfquery code with something completely different. So I took a look at the cfquerys that were pointed to the MS Access DBs and found the name of the "datasource" attribute. Using this attribute I went to the CF administrator on the 64 bit server and I added the web servivce WSDL file as a registered web service with an alias matching the datasource.

For example, if the datasource name was "AccessProxy", I opened the CF administrator, navigated to "data and services" and clicked on "Web services" and entered something like this:


Now, instead of a datasource named "acessProxy" I have a web service alias named "accessProxy".

### Simple Custom Tag

What's next? Again, the goal is to avoid too many code changes. I will have to make *some* changes, but I want it to be simple and straightforward. So my next step is to create a custom tag to replace the "cfquery" tag.

```
<!--- web service name --->
<cfparam name="attributes.datasource" default=""/>
<!--- query to return --->
<cfparam name="Attributes.name" default=""/>

<Cfif thistag.executionmode IS 'end'>
    <!--- trap the query string --->
    <cfset qString = thistag.generatedcontent/>
    <!--- create my WS obj --->
    <cfset accessObj = createobject("webservice","accessProxy")/>
    <!--- pass the query string --->
    <Cfset qry = accessObj.runQ(dsn=attributes.datasource, qString=qString)/>
    <!--- set the query back to name --->
    <cfset caller[attributes.name] = qry/>
    <!--- make sure nothing goes to the output buffer --->
    <Cfset thistag.generatedcontent = ''/>

</CFIF>
```

Let's review a couple of things. The name of the web service and the DSN need to be

the same on both servers. It is serving as an alias for my proxy and as a cfquery attribute on the 32 bit side. That's important because it will help me avoid too many code changes. It's also important that this custom tag go into the customTags directory - or a custom directory specified in CF Admin. Otherwise you will have to copy it around to folders where you want it to run. Finally, if possible you will want to name the file "query.cfm" - again to minimize code changes (as you shall see in a moment).

### Last Step, Alter the Cfquery

Now that I have my web service and custom tag in place, what is the final step? What changes do I need to make to my query to get it running through the proxy web service? The only change you will need to make is the addition of an underscore between CF and query. Your old Access query looked like this:

```
<cfquery name="users" datasource="AccessDSN">
    SELECT     *
    FROM    users
    WHERE     username = '#userName#'

</cfquery>
```

...And your new query looks like this:

```
<cf_query name="users" datasource="AccessDSN">
    SELECT     *
    FROM    users
    WHERE     username = '#userName#'

</cf_query>
```

### One Rather Large Caveat

You may have spotted it already. If you are using CFQUERYPARAM (and you should be) then you will have more code changes to make. In fact, you may wish to further enhance this proxy process and utilize an argument array or something for security reasons. In this case all my stuff is "export" stuff and quite old, and therefore (and this is not an excuse) it had very few cfqueryparams to contend with.

Whew! This entire process took longer to write up as a post than it did to create. Now I will await the judgment of my savvy and too smart for their britches readers who will explain to me how I *might* have done it if I had been as clever as they. Fire away :)