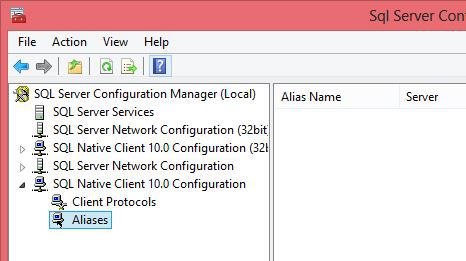
**[Aliasing a SQL Server: When it works, when it doesn’t and when it may be your problem.](http://www.sqlservercentral.com/blogs/sqlstudies/2014/06/18/aliasing-a-sql-server-when-it-works-when-it-doesnt-and-when-it-may-be-your-problem/)**

**FROM:** [**http://www.sqlservercentral.com/blogs/sqlstudies/2014/06/18/aliasing-a-sql-server-when-it-works-when-it-doesnt-and-when-it-may-be-your-problem/**](http://www.sqlservercentral.com/blogs/sqlstudies/2014/06/18/aliasing-a-sql-server-when-it-works-when-it-doesnt-and-when-it-may-be-your-problem/)

Creating an alias for a SQL Server is fairly easy and there are several ways to do it. Configuration Manager is my personal favorite. Open up configuration manager and select the SQL Native Client xx Configuration. Under that you will find Aliases.

[](https://sqlstudiesdotcom.files.wordpress.com/2014/06/alias1.jpg)

From here you can add, update or delete aliases.

So at this point some of the more polite members of the audience are probably thinking “Unfortunately I have no idea what you are talking about. Would you please explain what an alias is?” And I appreciate that from both of you. The rest are probably thinking something along the lines of “You idiot, if you’re going to talk about aliases it would be nice if you explained what they are first!”

Per [BOL](http://technet.microsoft.com/en-us/library/dd981072(v=sql.105).aspx):

An alias is an alternate name that can be used to make a connection. The alias encapsulates the required elements of a connection string, and exposes them with a name chosen by the user.

In other words once you have created an alias you can connect to the aliased machine using the new name in the connection string. For example I create an alias GEORGE and point it to my (local)\sql2012 instance. I can now connect to my instance using either its correct name (local)\sql2012 or GEORGE.

**When it’s working**

Let’s say you have a series of servers: LARRY, MOE and CURLY. You decide to do a side by side upgrade of CURLY. You are going to move all of the databases on CURLY (a 2008 server) to SHEMP (a 2012 server) and then shut CURLY down. There is a bit of a problem however. The developers have told you in no uncertain terms they do not have time to find all the dozens of places that CURLY was hard coded into the various applications. We can solve this easily enough by creating an alias on the application server pointing the alias CURLY to the new server SHEMP. Now when the connection strings try to go to CURLY the alias says to go to SHEMP and everything continues to work.

**When it doesn’t**

Over the weekend you’ve moved the databases and the applications were all tested. Everything went smoothly so CURLY was permanently shut down. Monday morning rolls around and the developers start calling. They can’t connect to the database. Want to guess why not? An alias only works locally, each individual client (the machine you want to connect from) must have its own alias created.

**When it might be your problem**

Over time alias’ get created (sometimes by accident believe it or not) and get forgotten. Over the years I have seen a number of situations where the answer to “Why can’t I connect to server XYZ?” is an alias that the user either didn’t know about or had forgotten. That’s why this has become one of those [Start with Stupid](http://sqlstudies.com/2014/03/17/start-with-stupid/) steps that I take when someone can’t connect but everything else looks ok.

**Comments**

http://www.sqlservercentral.com/Resources/Images/loading.gifLoading comments...

**Posted by Kenneth Fisher on 18 June 2014**

I have to admit I haven’t worked enough with 64bit systems enough to know about having to create the 2 aliases but that does make sense. I’m fairly sure about the creation of the alias on the DB Server though. I’ve tested that several times in the past and the alias only affects connections from that machine. IE you do in fact have to have aliases on each client even if you have on the DB server. You can of course create DNS aliases but that is something different and in my opinion much more powerful.

**Posted by Kenneth Fisher on 18 June 2014**

Got to love crazy security guys and even crazier server naming schemes!

**Posted by Andrew Notarian on 18 June 2014**

Anything driver-related is going to have 32- and 64-bit configurations (e.g. ODBC Configuration). Makes life interesting.

**Posted by Kenneth Fisher on 18 June 2014**

Yea I’ve run into the ODBC thing several times now. What a pain. I plan on doing a post at some point on creating ODBC connections for Access since I see that done wrong a lot.

**Posted by Sam Alex on 18 June 2014**

The problem we often see is when we have a named instance like MyServer/SQL2008 and we need to move the databases to MyNewServer/SQL2012 yet no one wants to make changes to the applications. I’ve not been able to get aliasing to work with an instance name in the alias name, and using instances rules out doing this through DNS.

Also someone commented that making the change on the server should be all that’s needed, but in my experience it has to be done via CLICONFG on all clients using the alias. The only time I create the alias on the server is if the server needs to reference itself as the alias name, which can happen depending on the situation.

**Posted by Kenneth Fisher on 18 June 2014**

Yea I haven’t been able to get Aliases of any type to work with instance names.

And yes as far as I can tell you have to create aliases of this type on every client. DNS aliases of course are a different case.

**Posted by Andrew notarian on 18 June 2014**

Once again, back in the old days before the browser service, I’d set up the other instance on a different port (say 2433) and then make a tcp/ip alias to it to address that issue.

**Posted by Kenneth Fisher on 18 June 2014**

You can still do that actually but it still has the problem of the instance name.

I’ve set up several instances on the same server with different IPs one pointing to port 1433 for each instance and can reference them as if they are default instances on the correct IP’s alias.

**Posted by Henrik Bernhard on 20 June 2014**

“When it doesn’t” – then deploy the alias via a gpo, and it’ll work :)  
(for those of us fortunate enough to be able to do this ourselves, or have connections in the AD team)

<http://blogs.msdn.com/b/sqlro/archive/2009/06/16/how-to-deploy-sql-server-client-aliases-using-active-directory-gpo-adm.aspx>

Should however be aware that the policy will overwrite all existing alias config on the targetted servers, so better add these to the policy before deploying it – or exclude the servers via group policy security filtering.

Can use a bit of scripting to check which servers have alias’es configured. Code below should do the trick. (configure the ‘PREFERRED DC’)

$logfile = “C:\temp\log.txt”  
$logfile32 = “C:\temp\log32.txt”  
Add-PSSnapin q\* -ErrorAction SilentlyContinue  
Connect-QADService ‘PREFERRED DC’  
$servers = Get-QADComputer -ComputerRole Member -OSName \*Server\*  
foreach ($server in $Servers)  
{  
$strMachineName = $server.name  
if (Test-Connection $strmachinename -quiet)  
{  
$objReg = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey(‘LocalMachine’, $strMachineName)  
$objRegKey= $objReg.OpenSubKey(“SOFTWARE\\Microsoft\\MSSQLServer\\Client\\ConnectTo” )  
$objRegKey32= $objReg.OpenSubKey(“SOFTWARE\\Wow6432Node\\Microsoft\\MSSQLServer\\Client\\ConnectTo” )  
$strMachineName  
#list 64/32 bit variables if they exist and dump in logfile  
IF($objRegKey) {Write-Host $strMachineName “64 bit”;$objRegKey.GetValueNames();$strMachineName >> $logfile;$objRegKey.GetValueNames() >> $logfile}  
IF($objRegKey32) {Write-Host $strMachineName “32 bit”;$objRegKey32.GetValueNames();$strMachineName >> $logfile32; $objRegKey32.GetValueNames() >> $logfile32}  
# clear variables  
$objRegKey = $Null  
$objRegKey32 = $null  
}  
ELSE  
{ write-host “Could not connect to server $strmachinename” -ForegroundColor Red }  
}