

# Re: Syntax

---

*Source:*

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.dts/2008-05/msg00070.html>

---

- *From:* thejamie <[thejamie@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:thejamie@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 29 May 2008 16:59:00 -0700
- 

It doesn't seem to come together well. I have a working package for a single country. It sets SQL Server (database.dbo.tablename) as the source connection in the dataflow task. There is an intermediate data conversion task. There is a destination which in this case has the same name as the tablename above except that it is an mdb rather than a table in SQL server.

For example tablename.mdb. Since it is working and I want it to work for all 250 tables, it seems logical to pull open a "FOR EACH" control so that is my first step in the Control Flow tab. The next step is to take the working dataflow task and cut it from the section and paste it into the For Each control.

The rest should be easy. Create a variable that references each of the mdb files that by definition have the same name as the table in the source sql server database. Thus each variable name is both the name of the table (plus the path of the mdb) and the datasource name of the mdb file.

Clicking on the sourceConnectionOLEDB and changing the source from the actual table name to the variable makes sense so that is what I did there. Thus my connection table is now the mdb file name minus the ".mdb" extension and the path.

Similarly, the Access mdb connection string should be the variable name as the database and the provider should be the Jet 4.0.

What would make sense to me is that once these criteria above are set up properly in the expressions for the source and destinations that executing the flow would work.

My syntax from what you indicate, is okay. I still get the same errors because the variable name does not appear in the evaluate expression – since this is the case, I am guessing that I won't see the name unless the for each loop is operating so I set the evaluate to happen at runtime.

Dunno why it should be any more complicated that this. The package already runs correctly so anything else would just break the package.

It seems to me that the variable itself does not pass properly through the

## Re: Syntax

for each loop and probably for debug purposes, Microsoft could write the For Each package so that the first variable in the list is persisted during the writing phase and will only not persist during runtime. At least this way, I could see whether or not my syntax is correct. I don't see anything different in what you have explained that is different from the many scenarios I have already tried. I'd like to be able to hit evaluate and actually see the name of the first mdb in the variable list. It would sure help in debugging this process.

--

Regards,  
Jamie

"jhofmeyr@xxxxxxxxxxxxxxxx" wrote:

Hi Jamie,

Let me see if I can help you save your arm some more :)

<quote>

I am still confused about why the double quotes are used at all in SSIS when working with the SQL Server connections.

</quote>

SSIS uses a VB.Net for scripting, and a VBA-like language for Expressions (including Derived Column expressions). This VBA-like language uses double quotes (") to delimit strings. What this means, is that when you're building a SQL Statement inside an Expression, you need to enclose the string portion in double quotes, and the SQL string portions in single quotes (') as T-SQL uses single quotes as string delimiters. So in short – the SQL statement SELECT 'Hello' in an expression would be: "SELECT 'Hello'". This holds true for when you need to substitute a package variable for any portion of the statement. Imagine you need to substitute in the table name and code value in the following statement SELECT \* FROM table1 WHERE code = 'testcode' you expression would look something like: "SELECT \* FROM " + @[User::sTable] + " WHERE code = " + @[User::sCode] + "" Another thing to remember is that the escape character for the VBA-like string special characters is the backslash ("\") – so when you're constructing a path, you will need to double-up on backslashes (so "C:\Test.mdb" becomes "C:\\Test.mdb"). I don't know if that clarifies things :-/

<quote>

in the case of Access, the connection string is actually just the filename and path

</quote>

This is not entirely accurate. I created a package and OLEDB connection to an Access DB to double check, and the connection string ended up as:

"Data Source=C:\db1.mdb;Provider=Microsoft.Jet.OLEDB.4.0;" Clearly

## Re: Syntax

this contains the "X=Y" format that it's asking for :)

<quote>

So, you mention to be careful regarding the "'s (single quotes) which

....

</quote>

Hopefully this has been adequately explained above. When you type in a SQL statement directly, you use single-quotes for strings as you are simply typing SQL code. When you use an Expression to build the code, you quote and substitute variables the same way as you would inside (for example) a VB application

<quote>

The connection string format is not valid.

....

</quote>

See Above

<quote>

.... wouldn't that be the "VALUE" portion of the @[User::TargetDB] loop?

</quote>

Not if the "[User::TargetDB]" appeared \*inside\* the Expression string quotes (" ... @[User::TargetDB]..."). In this case SSIS will just assume it is part of the string.

<quote>

.... 16 clicks required to run the wizard once for each of the 250 plus Access databases by hand.

</quote>

Ouch!

Good luck!

J