SQL Server Integration Services

SSIS & WMI Data Reader Task Query

Integration Services has a huge range of features on offer, but is often seen only as an ETL tool for data warehouse projects. It can however be much more useful in a wide range of scenarios. One of the least used (and least covered) features of Integration Services is the Windows Management Instrument (WMI) Data reader task and Event Watcher Task. WMI can tell you pretty much anything about a machine, from memory to application event log errors and everything in between.

This article will cover how we can make use of the WMI Data Reader Task to retrieve information about remote machines. Our example will focus Disk space usage of remote machines.

The problem:
We need to know how much disk space is available on a selection of servers (or PC's)

The Solution:
We will use WMI Query Data Reader Task to query the remote machines available disk space.

Before we begin:
To query a remote machine you must have Administrator privileges on the remote machine or run the package as an Administrator. In most scenarios this will mean running the package as a network administrator or supplying the relevant Administrator logon credentials.

One Caveat to be aware of here is that you cannot query the local system, using supplied credentials. (The local machine must be queried using Windows Authentication). This example assumes you are only interested in remote machines to keep things relatively simple and easy to follow.

Step 1:
Setup a database name **wmi_query_db**.

Step 2:
Create the server list table:

```
-- Server Query List
CREATE TABLE [dbo].[server_query_list](
        [server] [varchar](max) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
```

(Once the table is create enter the remote address of some servers, supplying the domain details e.g.: myPC.myNetwork.com)

Step 3:
Create the table in which to store the details: Server Name, Device ID [Drive], Description, Size, Free Space and QueryDT.

```
USE [wmi_query_db]
GO
/****** Object:  Table [dbo].[ServerDiskSpace]    Script Date: 06/26/2009 11:36:36 ******/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ServerDiskSpace](
	[ServerName] [varchar](50) NULL,
	[DeviceID] [varchar](50) NULL,
	[Description] [varchar](50) NULL,
	[Size] [varchar](50) NULL,
	[FreeSpace] [varchar](50) NULL,
	[QueryDT] [datetime] NULL
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
```

Now we have our database and the relevant tables for a query list and somewhere to put the details can create our package.

Step 4:
Create a SSIS package – **WMI_Query**

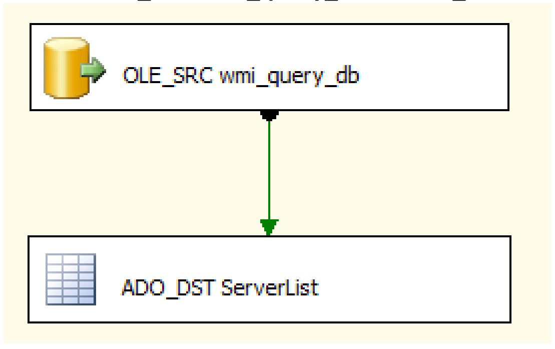
Package Outline:
The package we will create will, for each server in our table **server_query_list,** connect to the server, query the disk space and return the details to the **ServerDiskSpace** table

Step 5 – Getting the list of server to query:
This step will retrieve the list of servers from the database that we want to query.

1. Create a Package level variable as described:

Variable Name:	ServerList
Scope:	{replace with your package name}
DataType:	Object

   (This will hold the list of servers that we which to query in our Package).
2. Create a Data Flow Task in the Control Flow work area. And click onto the Data Flow work Tab
   a. Create an OLE DB source connection to the **server_query_list** table in the **wmi_query_db**. Name it **OLE_SRC wmi_query_db**
   b. Now add a **Record Destination** to the data flow work area. Name it **ADO_DST ServerList**
   c. Connect **OLE_SRC wmi_query_db** to **ADO_DST ServerList** and map the server



   d. Double-Click **ADO_DST ServerList.**

e. On the **Component Properties** tab under custom properties enter the variable name **User::ServerList** (that we created in #1.)

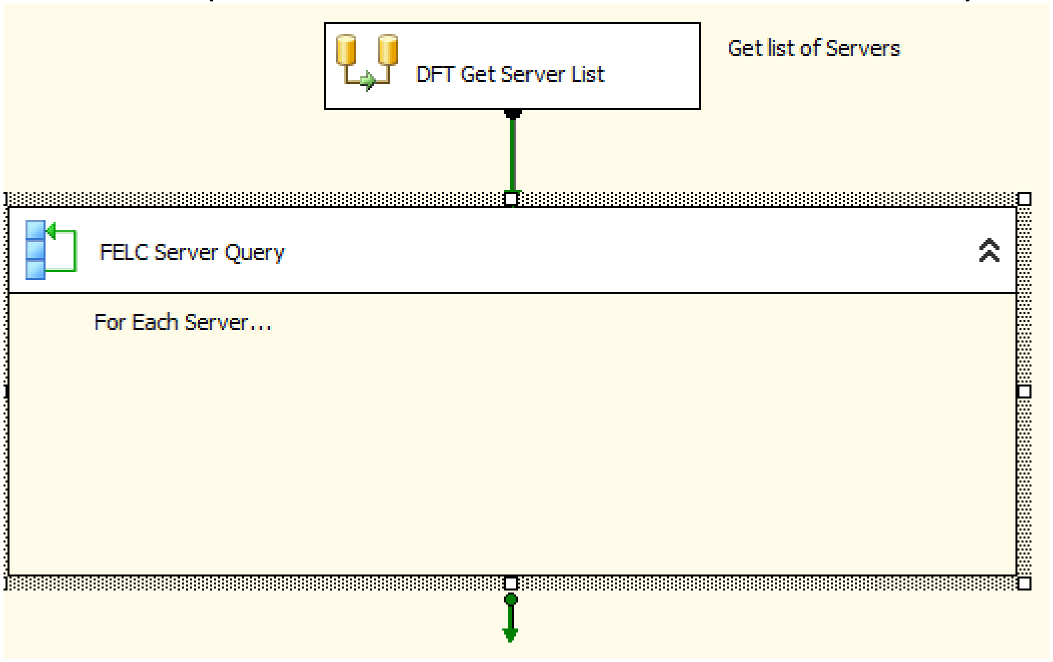
f. Click ok

The servers will now be stored in a ADO record set, that we will use in the following step.

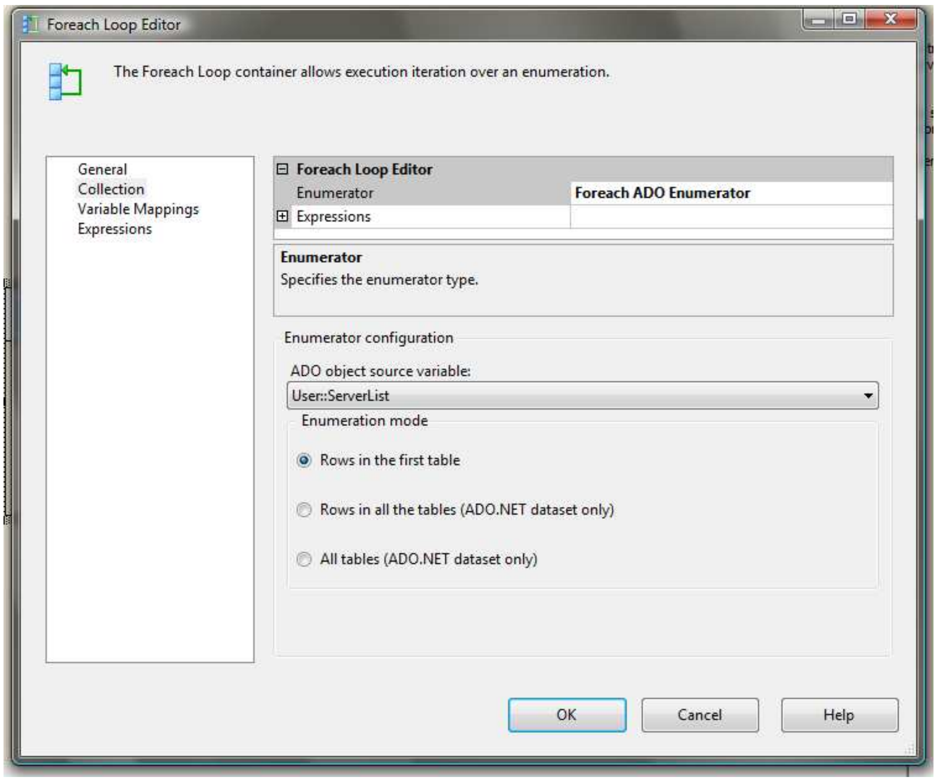
Step 6 – Building The Server Query Loop

Now we have server to query, we can build the loop to iterate through the server list.

3. Add a **For each Loop Container** to the data flow work area and name it **FELC Server Query**



4. Double-Click the **For Each Loop Container** and select **Collection**
5. On the **Collection** form,
   a. set the ADO Object Source Variable to **User::ServerList**.
   b. Select the Enumeration mode to **Rows in the first table**


Foreach Loop Editor

The Foreach Loop container allows execution iteration over an enumeration.

Foreach Loop Editor	
Enumerator	Foreach ADO Enumerator
Expressions	

Enumerator
Specifies the enumerator type.

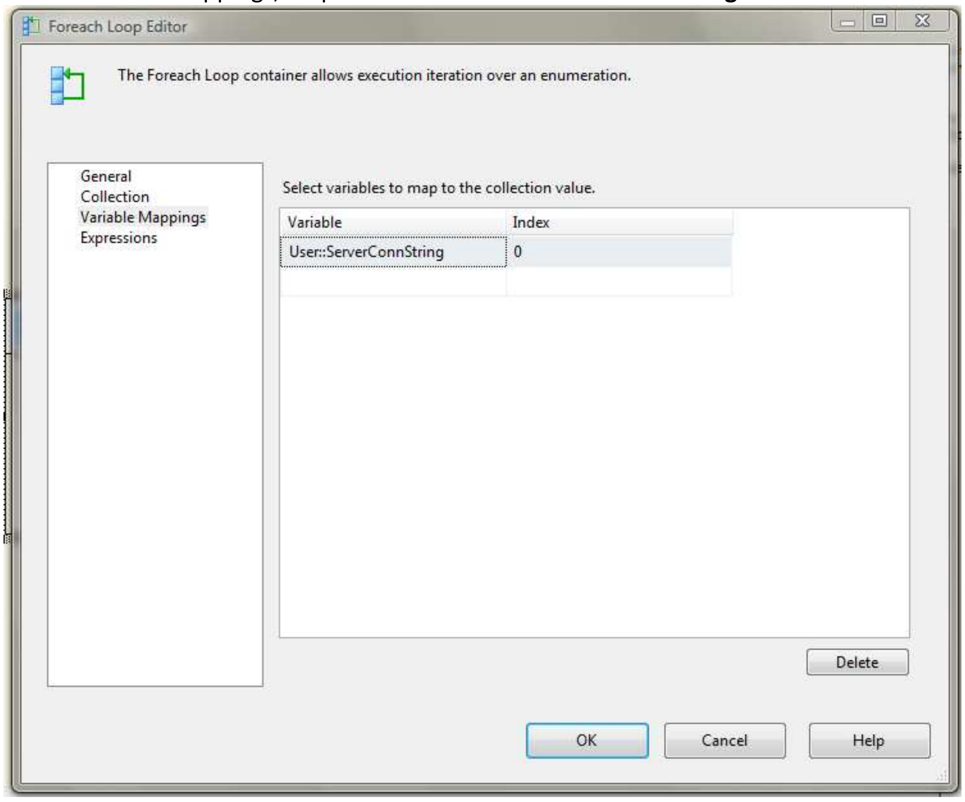
Enumerator configuration

ADO object source variable: User::ServerList

Enumeration mode

- [✓ Rows in the first table]
- Rows in all the tables (ADO.NET dataset only)
- All tables (ADO.NET dataset only)

c. Under Variable mappings, map the Variable **User::ServerConnString** to index **0.**


Foreach Loop Editor

The Foreach Loop container allows execution iteration over an enumeration.

Select variables to map to the collection value.

Variable	Index
User::ServerConnString	0

d.

6. Click ok.

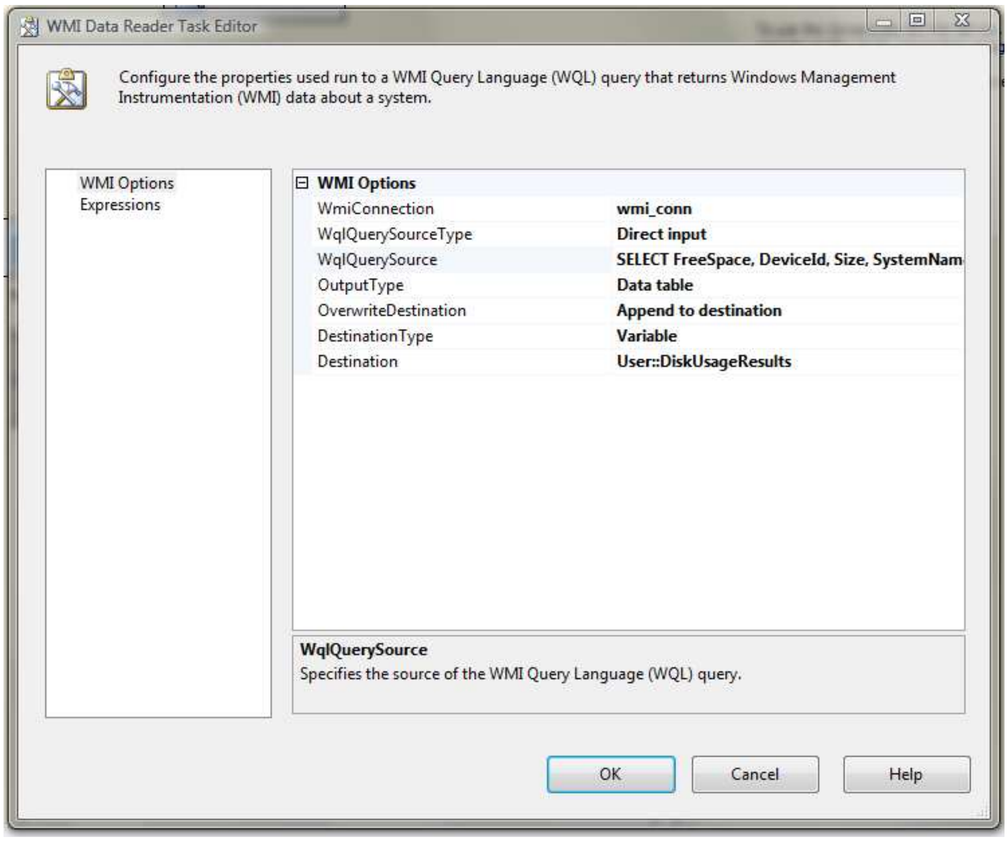
Step 7 – The WMI Query!
Here we specify what we want to query on our servers.

1. Within **FELC Server Query** container Add a **WMI Data Reader Task**
2. Define a new variable DiskUsageResults (this is where the returned information will be stored in the first instance):

Variable Name:	DiskUsageResults
Scope:	{replace with your package name}
DataType:	Object

3. Double-click the **WMI Data Reader Task** and go to properties
   a. On WMI Options, set the properties as follows:

WmiConnection:	wmi_conn (When you define a new WMI connection, Leave all settings as per defaults – we will change these in the looping process)
WalQuerySourceType:	DirectInput
WqlQuerySource:	SELECT FreeSpace, DeviceId, Size, SystemName, Description FROM Win32_LogicalDisk where DriveType= 3
OutputType:	DataTable
OverwriteDestination:	Append to destination
DestinationType:	Variable
Destination:	User::DiskUsageResults



Configure the properties used run to a WMI Query Language (WQL) query that returns Windows Management Instrumentation (WMI) data about a system.

WMI Options	
WmiConnection	wmi_conn
WqlQuerySourceType	Direct input
WqlQuerySource	SELECT FreeSpace, DeviceId, Size, SystemNam
OutputType	Data table
OverwriteDestination	Append to destination
DestinationType	Variable
Destination	User::DiskUsageResults

WqlQuerySource
Specifies the source of the WMI Query Language (WQL) query.

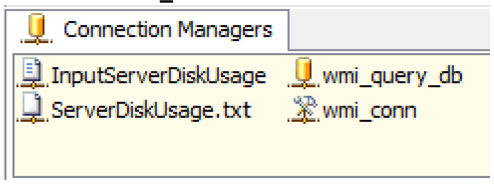
We have now defined our basic WMI Query. (this would query the local machine if we done nothing else).

4. Change the name of the WMI Data Reader task to **WMI_DRT Server Disk Usage (**This makes it easier to track in package progress)

Step 8 - The Remote Machine WMI Connection
Before we can run the WMI query against another though machine we need to set the connection to the remote machine by using an expression for the connection.

Note: This step of the process is pivotal to the process working.

1. Select the **wmi_conn** I connection from the connection manager pane



2. In the properties pane, find and expand the **Expressions** property
3. Click the eclipses to set the connection string expression.

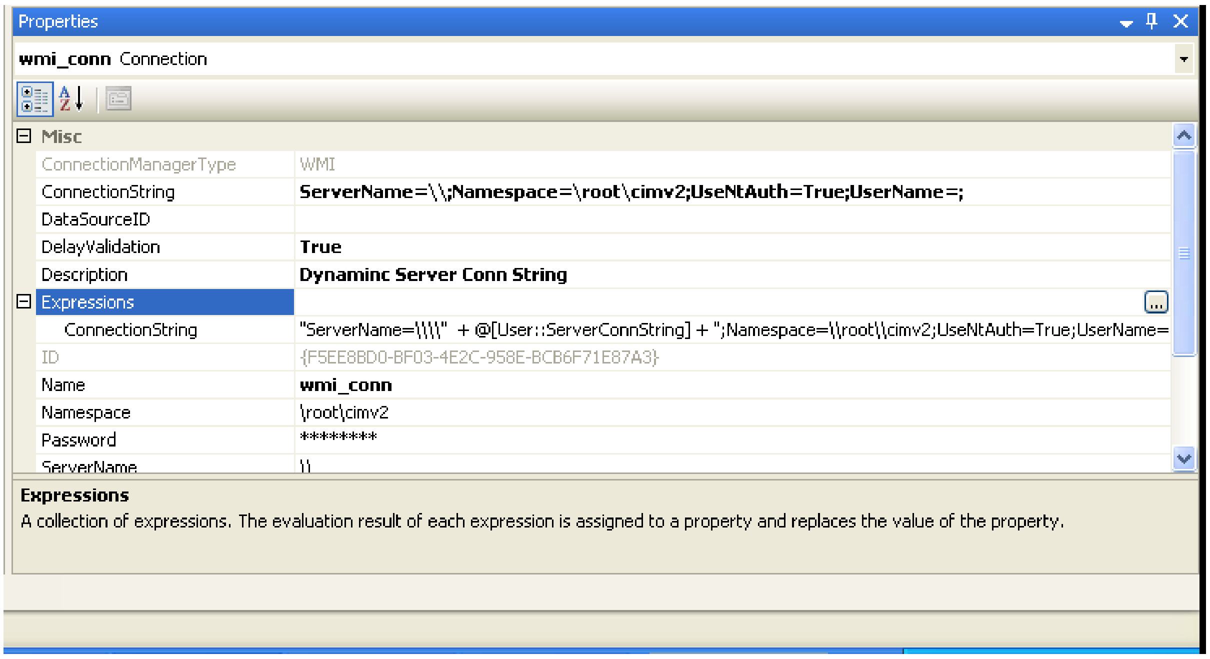
   Now, if this will be run by a user account will admin rights to the remote machine, you could use windows authentication:

   ```
   "ServerName=\\\\"  + @[User::ServerConnString] +
   ";Namespace=\\root\\cimv2;UseNtAuth=True;UserName=;"
   ```

   If you which to supply the logon credentials of the account with admin rights to the remote machines, you must you the following connection string, supplying the UserName and PassWord:

   ```
   "ServerName=\\\\"  + @[User::ServerConnString] +
   ";Namespace=\\root\\cimv2;UseNtAuth=False;UserName=Administrator;PassWord=XXXXXXX"
   ```

   This result should look like this:



Properties

wmi_conn Connection

Misc	
ConnectionManagerType	WMI
ConnectionString	**ServerName=\\;Namespace=\root\cimv2;UseNtAuth=True;UserName=;**
DataSourceID	
DelayValidation	**True**
Description	**Dynaminc Server Conn String**
Expressions	
ConnectionString	"ServerName=\\\\" + @[User::ServerConnString] + ";Namespace=\\root\\cimv2;UseNtAuth=True;UserName=
ID	{F5EE8BD0-BF03-4E2C-958E-BCB6F71E87A3}
Name	**wmi_conn**
Namespace	\root\cimv2
Password	********
ServerName	\\

Expressions
A collection of expressions. The evaluation result of each expression is assigned to a property and replaces the value of the property.

Important Considerations:

First of all, if you intend to run the package as a user with admin rights using Windows Authentication, then the easy way to do this is by creating a Security credential in SQL server and a SQL Server Agent, SSIS Package Execution proxy to run the job as a user with right on a scheduled basis.

Secondly. If you use windows authentication, why not just set the server name? Well in short SSIS has a bug and it can't be done. (**https://connect.microsoft.com/SQLServer/feedback/ViewFeedback.aspx?FeedbackID=377218**)

Thirdly, if you intend not to use windows authentication and run the package by entering the username and password, you should ensure that you use package protection to protect user account details of the admin rights user account, as the password is entered in clear text format as you are constructing the string.

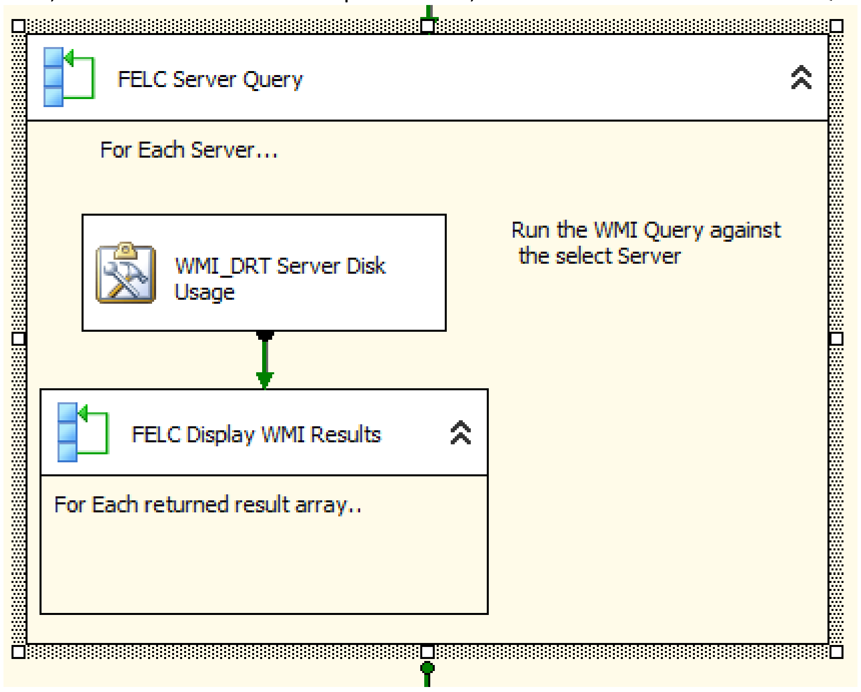
Step 9 – Returning the results

Once we can connect to the server and run our WMI Query, we need to return the result set to our database. Now to do this, we need to return the results to variable as an ADO record set and then read through the table to insert the results into our database table.

1. Define the following package variables:

Variable Name:	**Scope:**	**DataType:**
FreeSpace	DiskUsageResults	String
DeviceID	DiskUsageResults	String
Description	DiskUsageResults	String
SystemName	DiskUsageResults	String
Size	DiskUsageResults	String

2. Next, Add another For Each Loop Container, nested within the **FELC Server Query.** Name it **FELC WMI Query Results**

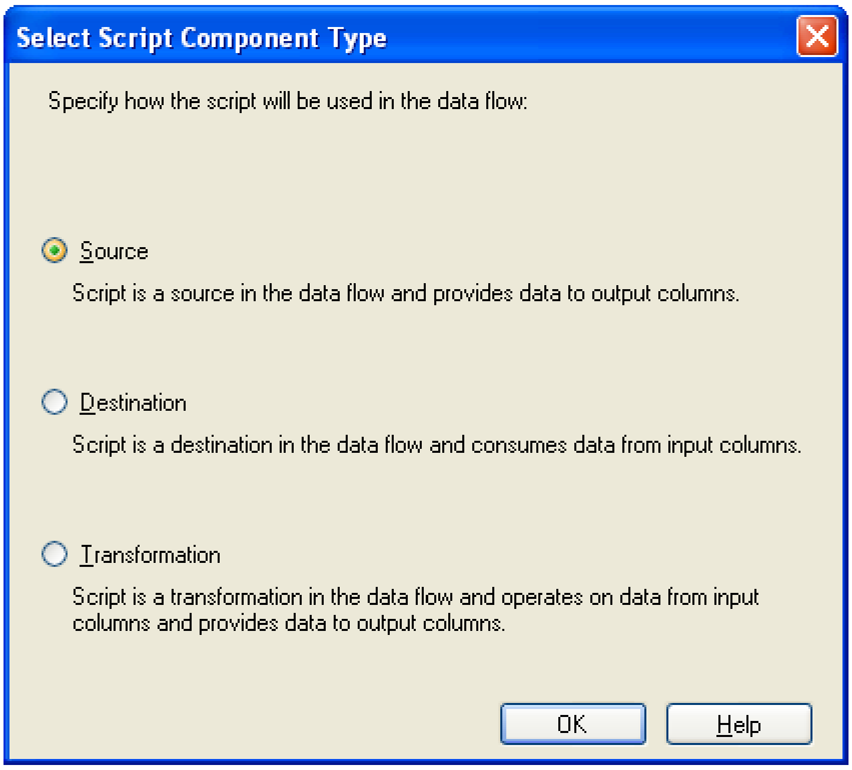

FELC Server Query
For Each Server…
WMI_DRT Server Disk Usage
Run the WMI Query against the select Server
FELC Display WMI Results
For Each returned result array..

3. Connect the WMI Data Reader Task to the **FELC WMI Query Results:**
   a. In the properties of the nested for each loop container (**FELC WMI Query Results)**, go to collections and specify the ADO Object Source Variable as **User::DiskUsageResults** (as defined in **Step 7** - 2)
   b. Set the Enumeration Mode as: **Rows in First Table**
   c. Under Variable mappings, map the Variable as follows:

Variable	**Index**
User::Description	0
User::DeviceID	1
User::FreeSpace	2
User::Size	3
User::SystemName	4

4. Next, within the **FELC WMI Query Results** container add another data flow task and name it: **DFT Export Results**

5. Within the data flow add a **Script component task,** and specify it as a **Source.** Click ok.


Select Script Component Type

Specify how the script will be used in the data flow:

[● Source]
Script is a source in the data flow and provides data to output columns.

○ Destination
Script is a destination in the data flow and consumes data from input columns.

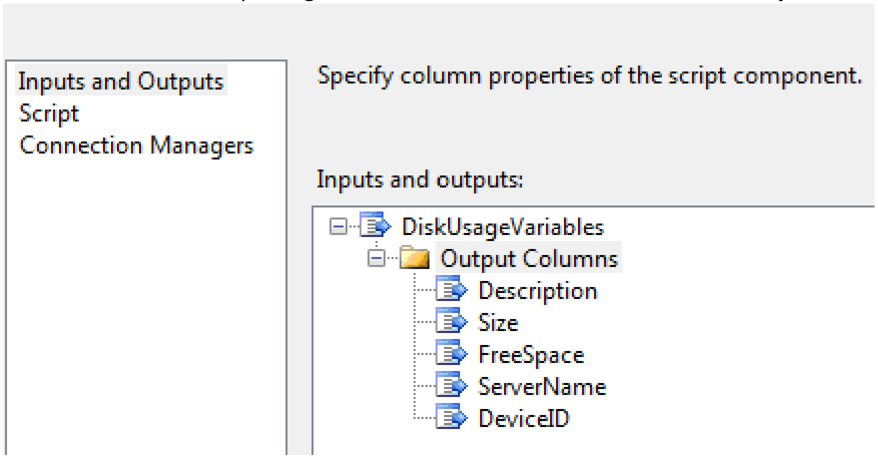
○ Transformation
Script is a transformation in the data flow and operates on data from input columns and provides data to output columns.

6. Now under Inputs and Outputs rename the output as **DiskUsageVariables**.

   Note: The output name is important in our script. So give it a sensible name without spaces.

7. Now under **Output Columns** define the Output columns for our retrieved data. Here I've set all data types to **string [DT_STR]** and a **length: 50**

   Note: It's not a necessity, but it can help to be consistent with naming. So I have created output column names to match those of the package variables we used in the **FELC WMI Query Results** collection mapping.



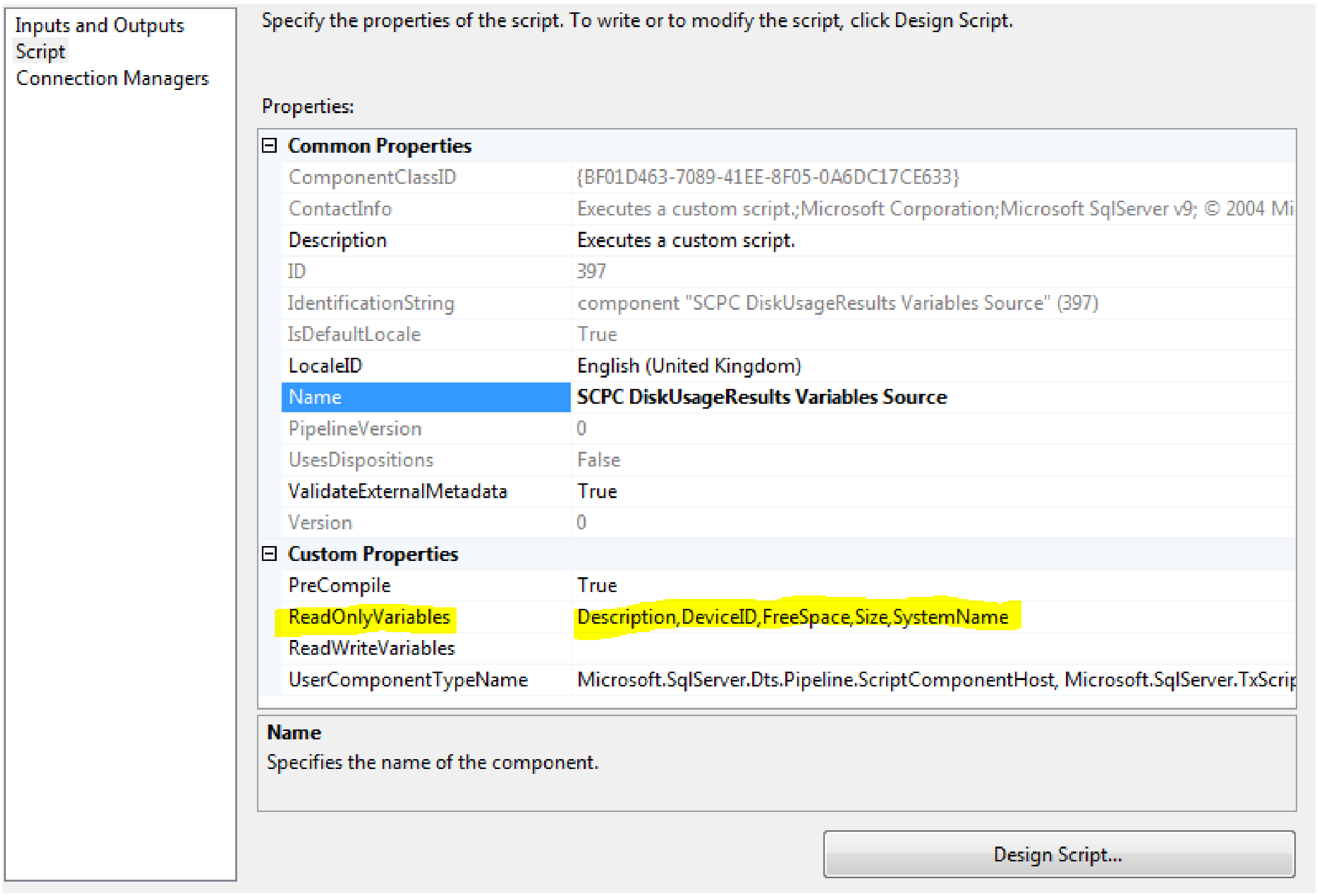
Specify column properties of the script component.

Inputs and outputs:

```
DiskUsageVariables
  Output Columns
    Description
    Size
    FreeSpace
    ServerName
    DeviceID
```

8. Next, select **Script** and add the variables names (separated by a comma, no spaces) to the **ReadOnlyVariables**

property:
Description,DeviceID,FreeSpace,Size,SystemName



Inputs and Outputs
Script
Connection Managers

Specify the properties of the script. To write or to modify the script, click Design Script.

Properties:

Common Properties	
ComponentClassID	{BF01D463-7089-41EE-8F05-0A6DC17CE633}
ContactInfo	Executes a custom script.;Microsoft Corporation;Microsoft SqlServer v9; © 2004 Mi
Description	Executes a custom script.
ID	397
IdentificationString	component "SCPC DiskUsageResults Variables Source" (397)
IsDefaultLocale	True
LocaleID	English (United Kingdom)
Name	**SCPC DiskUsageResults Variables Source**
PipelineVersion	0
UsesDispositions	False
ValidateExternalMetadata	True
Version	0
Custom Properties	
PreCompile	True
ReadOnlyVariables	Description,DeviceID,FreeSpace,Size,SystemName
ReadWriteVariables	
UserComponentTypeName	Microsoft.SqlServer.Dts.Pipeline.ScriptComponentHost, Microsoft.SqlServer.TxScrip

Name
Specifies the name of the component.

9. Now click the **Define Script** button, define the script as follows:

```
' Microsoft SQL Server Integration Services user script component
' This is your new script component in Microsoft Visual Basic .NET
' ScriptMain is the entrypoint class for script components

Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Pipeline.Wrapper
Imports Microsoft.SqlServer.Dts.Runtime.Wrapper

Public Class ScriptMain
    Inherits UserComponent

    Public Overrides Sub CreateNewOutputRows()
        '
        ' Add rows by calling AddRow method on member variable called "<Output
Name>Buffer"
        ' E.g., MyOutputBuffer.AddRow() if your output was named "My Output"
```

```
            '
            With DiskUsageVariablesBuffer
                .AddRow()
                .Description = Variables.Description
                .DeviceID = Variables.DeviceID
                .FreeSpace = Variables.FreeSpace
                .Size = Variables.Size
                .ServerName = Variables.SystemName
            End With
        End Sub
    End Class
```

Note: Again this is an important step to get right. The *with buffer* name must also match the output name you wish to refer to. In our example, this needs to be: `With DiskUsageVariablesBuffer`

10. Save and return to the Script editor and then click **ok** to exit the script editor.

11. Re-name this script component as **SCPC DiskUsageResults Variables Source** to give it a move useful name.

You're now on the final stretch!

To make this information more useful than a simple snapshot, you can now add additional information to data flow by using a derived column add details, such as the query date, which we will now do.

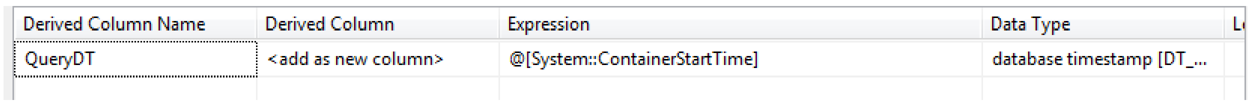
12. So, add to the data flow a **Derived Column** container and connect this to the **SCPC DiskUsageResults Variables Source** container.

13. Give the container a sensible name: **DER Query Date**

14. Double-Click the **Derived Column** container and define adds the package execution time to a derived column in the data flow called **QueryDT.**

Set the details as follows:

Derived Column Name:	QueryDT
Derived Column:	<add as new column>
Derived Column Expression:	@[System::ContainerStartTime]
Data Type:	Database timestamp [DT_DBTIMESTAMP]

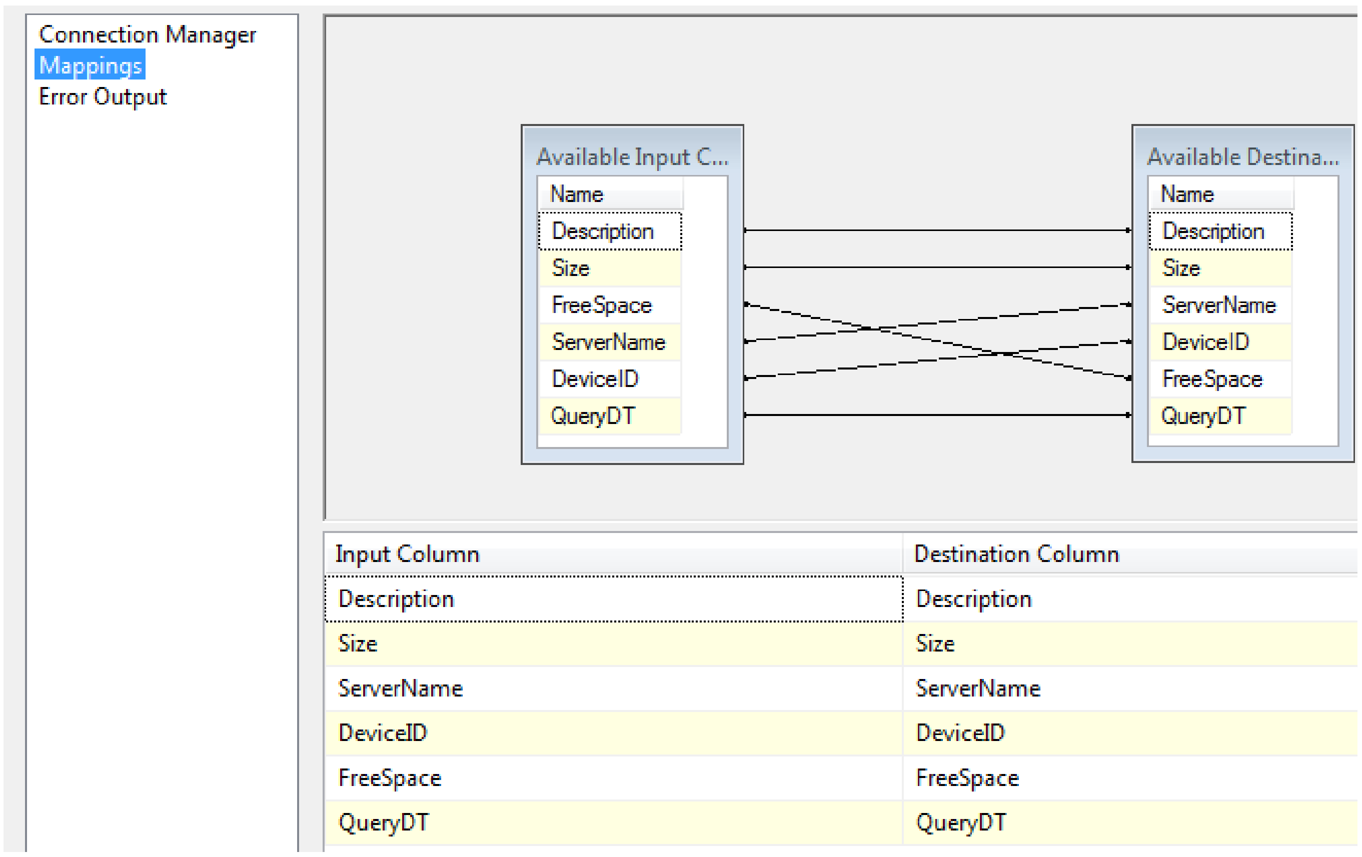


Derived Column Name	Derived Column	Expression	Data Type	L
QueryDT	<add as new column>	@[System::ContainerStartTime]	database timestamp [DT_...	

15. Last of all, add a **SQL Server Destination** to the data flow and connect the Derived data task to rename it .

16. Define the SQL Server Destination to connect the **wmi_query_db** and db table **ServerDiskSpace**

17. Under **Mappings** map the data flow columns to the table columns as follows:



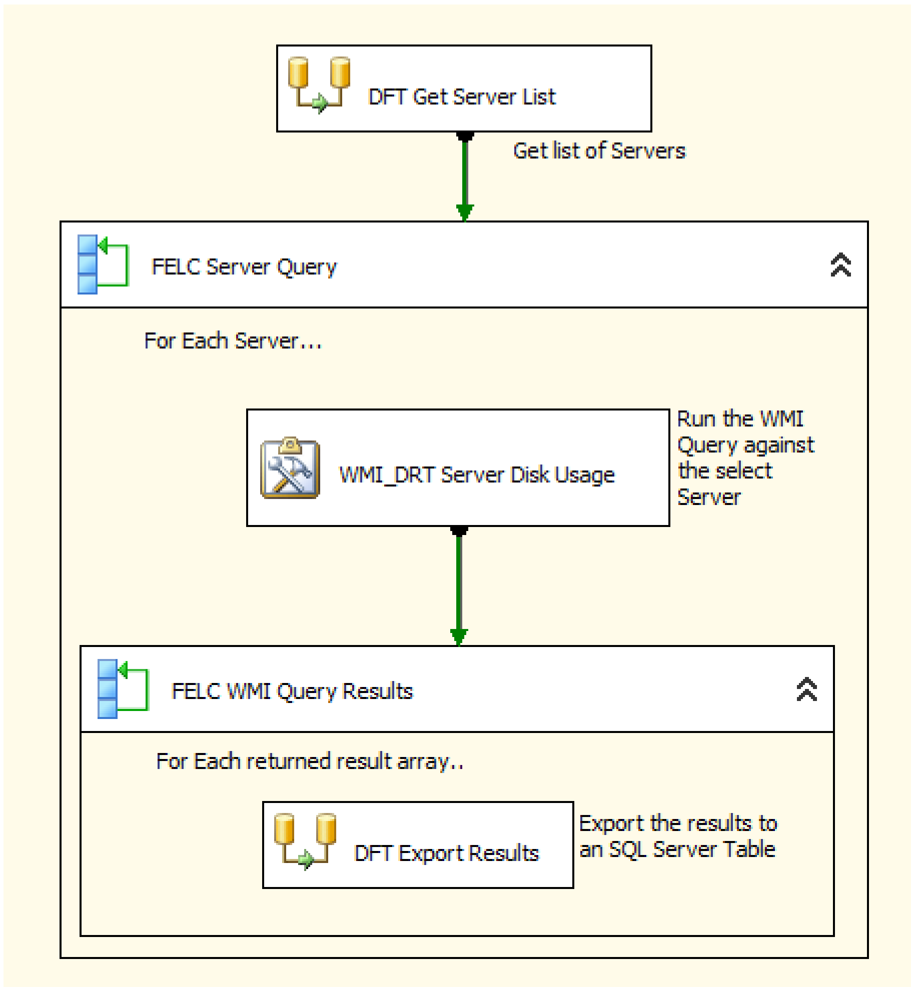
Input Column	Destination Column
Description	Description
Size	Size
ServerName	ServerName
DeviceID	DeviceID
FreeSpace	FreeSpace
QueryDT	QueryDT

18. Click Ok

Conclusion

That's it! You've Done! You have your package that:

1. Gets a list of servers from a database table
2. Queries each server in turn
3. Returns the query results to a database table


DFT Get Server List
Get list of Servers
FELC Server Query
For Each Server...
WMI_DRT Server Disk Usage
Run the WMI Query against the select Server
FELC WMI Query Results
For Each returned result array..
DFT Export Results
Export the results to an SQL Server Table

This is just one of many ways to use the WMI Query Data Reader. However, what this article does is provide you with the framework to create any query you like and return the results.

To adapt this process, first ask yourself the question, what do I want to know about the machine? Start by first establishing what your WMI Query will be. Microsoft provide a useful download for this: WMI Code Creator 1.0

From here you can build the relevant framework for retrieving and storing the information you need. The key modifications that would need to be made to retrieve other information about remote machines would be:

1. WMI Query
2. The package variables for retrieved information
3. FELC WMI Query Results – variable mappings
4. SCPS Package Variables Source variable – variables and buffer output descriptions.
5. Database storage table

The possibilities are endless..........