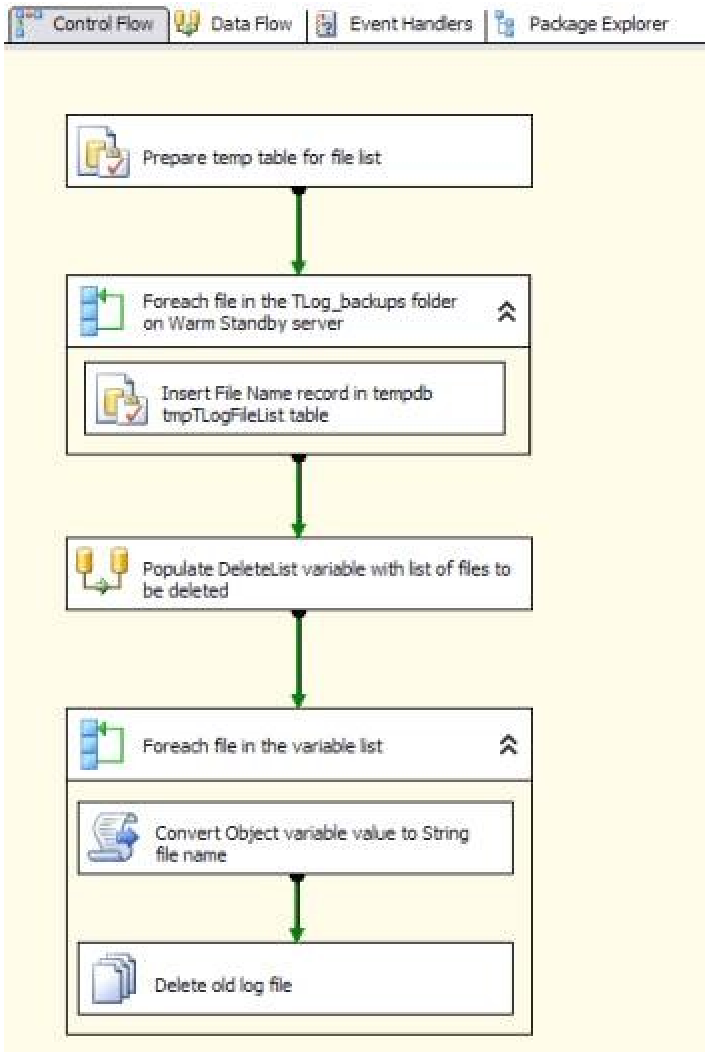
Real World Foreach Loop Container example

Looping operations in SQL Server Integration Services

The Foreach Loop container is one of the most flexible and useful controls available to SQL Server Integration Services developers, but the fact that it's relatively poorly documented means that it may not get used as often as it deserves.

In this example I'll describe how I used it to build a workaround for a problem in SQL Server log shipping. Due to low disk space we needed to delete log files as soon as they were no longer needed without changing the log maintenance configuration, but this had to be both automated and failsafe. The Foreach Loop container made this possible. You can adapt this solution for any situation where operations need to be regularly performed on an unknown number of files with no fixed filenames.

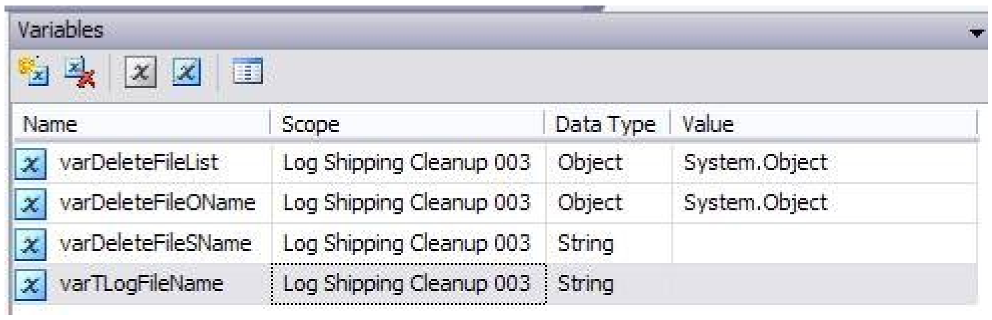
I'm assuming that you're familiar with using the SQL BI Dev Studio tools and building basic packages. If this isn't the case, I recommend working your way through the Integration Services Tutorial in SQL Books Online to become familiar with the basic concepts. Jamie Thomson's blog is an invaluable resource when you're ready to go further with SSIS.


Control Flow
Data Flow
Event Handlers
Package Explorer
Prepare temp table for file list
Foreach file in the TLog_backups folder on Warm Standby server
Insert File Name record in tempdb tmpTLogFileList table
Populate DeleteList variable with list of files to be deleted
Foreach file in the variable list
Convert Object variable value to String file name
Delete old log file

Connections and Variables

I defined two connections in a new SSIS package, a regular SQL Server data source pointing towards the SQL log shipping monitor/target server and an ADO.NET connection pointing towards TEMPDB on the target SQL server.

I also defined four package-scoped variables (as shown below). These will be explained in more detail as we go on.



Name	Scope	Data Type	Value
varDeleteFileList	Log Shipping Cleanup 003	Object	System.Object
varDeleteFileOName	Log Shipping Cleanup 003	Object	System.Object
varDeleteFileSName	Log Shipping Cleanup 003	String	
varTLogFileName	Log Shipping Cleanup 003	String	

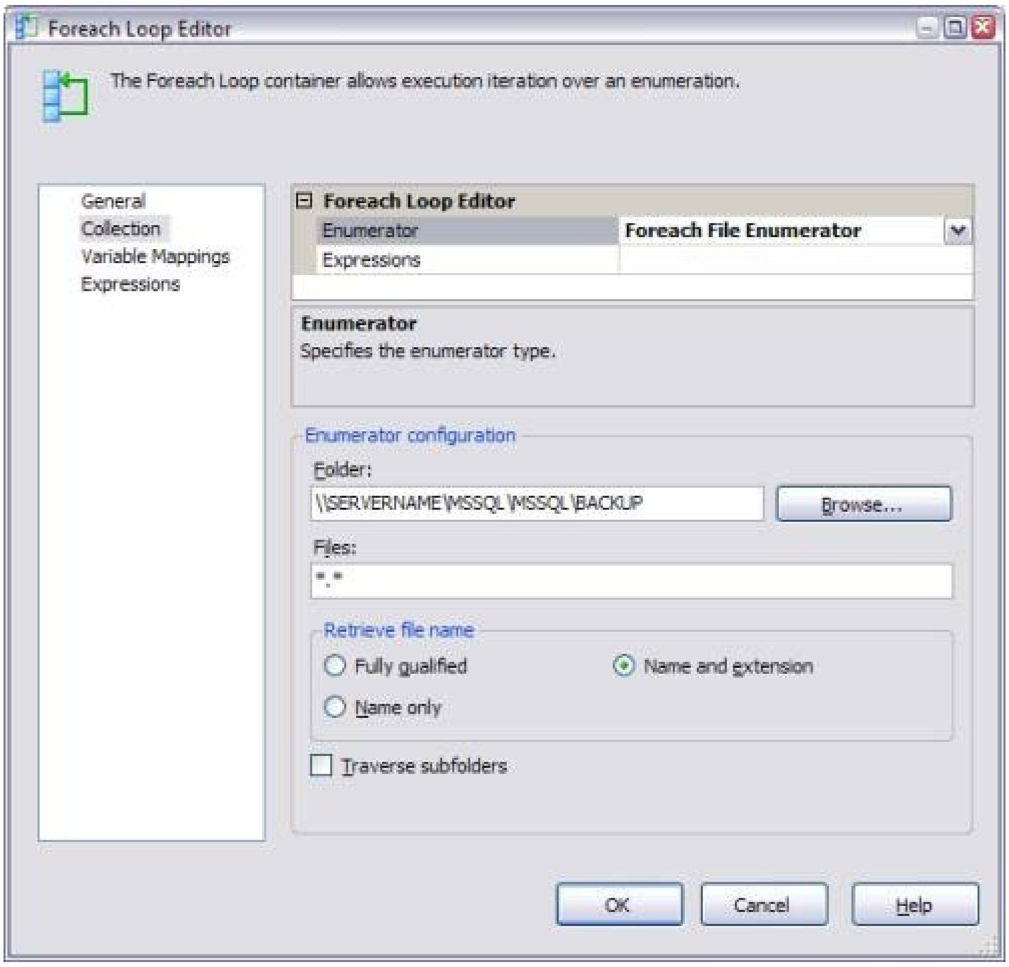
Setting the stage

The first step was an Execute SQL Task that creates a temporary table for the list of filenames we'll be handling. I used the TEMPDB connection and added this code to the SQLStatement property with a SQLSourceType of 'Direct input':

```
if not exists     (select *
            from  tempdb..sysobjects
            where xtype =     'U'
            and name =  'tmpTLogFileList')

create table tmpTLogFileList
      (id int identity(1, 1) not null,
      FileName nvarchar(128) not null)
else

truncate table tmpTLogFileList
```

Building the list of files

The next step was to populate this table with a list of files found in a given folder. To do this I added a Foreach Loop Container to the package. On the Collection tab I selected the 'Foreach File enumerator' and entered the path to the folder. I could have used a variable here, but in this case the path was fixed so that wasn't necessary. The 'Name and extension' option was also selected.



The Foreach Loop container allows execution iteration over an enumeration.

Foreach Loop Editor

Enumerator	Foreach File Enumerator
Expressions	

Enumerator
Specifies the enumerator type.

Enumerator configuration

Folder: \\SERVERNAME\MSSQL\MSSQL\BACKUP

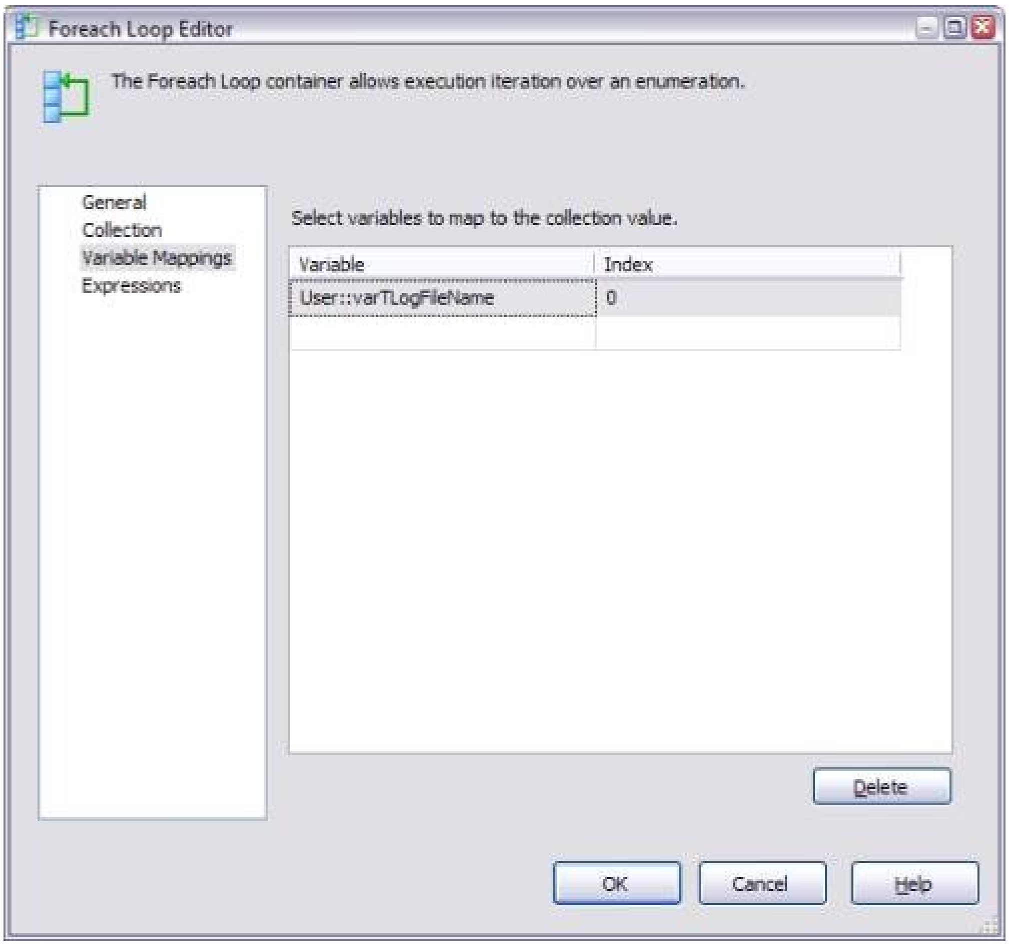
Files: *.*

Retrieve file name

Fully qualified [✓ Name and extension] Name only

☐ Traverse subfolders

On the 'Variable Mappings' tab I added a mapping to the **varTLogFileName** variable I defined earlier. The Foreach Loop will move through each file in the folder I defined and update this variable each time with each file name it encounters.



The Foreach Loop container allows execution iteration over an enumeration.

Select variables to map to the collection value.

Variable	Index
User::varTLogFileName	0

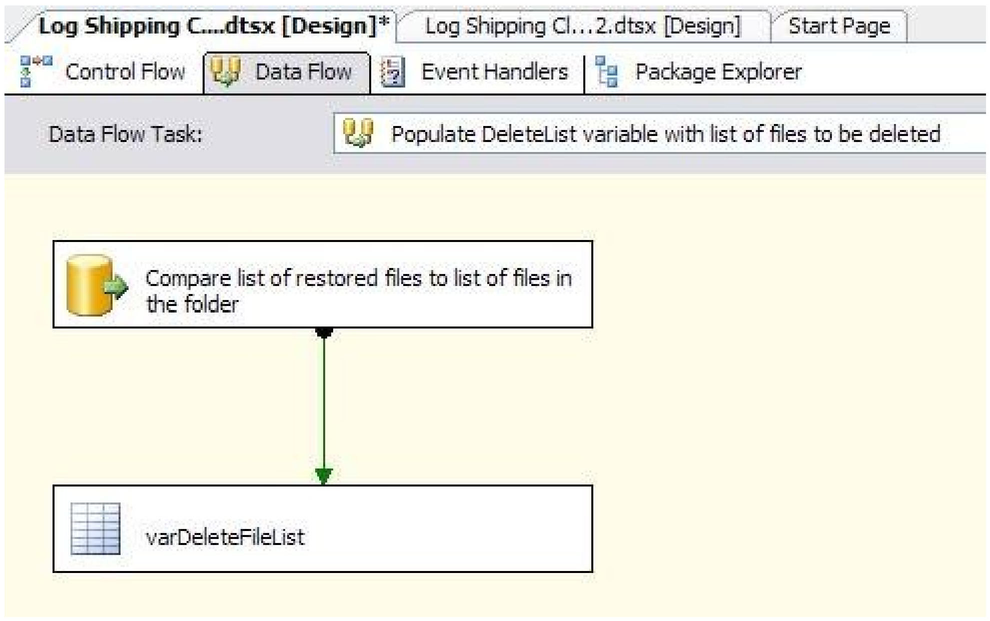
The container's properties were now correctly set. I then added a Execute SQL task within the container. This used the TEMPDB connection again with the following code:

```
insert into tmpTLogFileList (FileName)
      values(@FileName)
```

On the 'Parameter Mapping' tab I added the **varTLogFileName** variable as an Input parameter with a mapping to the **@FileName** variable used in the SQL command (as shown above). At runtime, the Loop container runs this command once for each filename and the tmpTLogFileList SQL table is populated.

Checking which files can be safely deleted

I now had a table with a complete list of all the files found in the folder, but in this particular case I needed to be sure that these backup files had been correctly restored on the log shipping target server before deleting them.


Log Shipping C....dtsx [Design]*
Log Shipping Cl...2.dtsx [Design]
Start Page
Control Flow
Data Flow
Event Handlers
Package Explorer
Data Flow Task:
Populate DeleteList variable with list of files to be deleted
Compare list of restored files to list of files in the folder
varDeleteFileList

I created a Data Flow task to build the list of files to be deleted. The source was an OLE DB connection to the log shipping target SQL server with a SQL query joining the tmpTLogFileList table with the system table used by SQL to store the restore status:

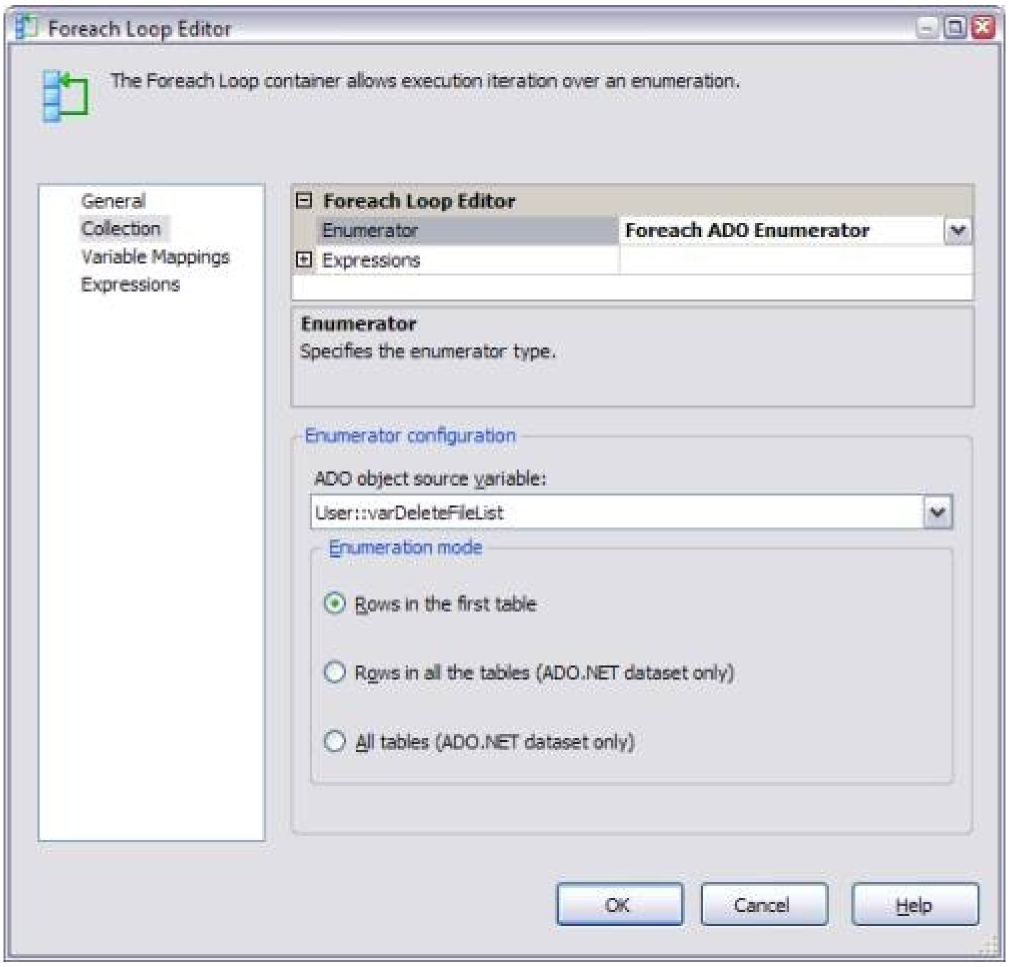
```
select      FileName
from   tempdb.dbo.tmpTLogFileList
where id <=
       (select id - 16
       from   tempdb.dbo.tmpTLogFileList
       where FileName in
              (select      max(last_file)
              from   msdb.dbo.log_shipping_plan_history
              where succeeded = 1
              and    activity =  1)
       )
and    FileName like '%.TRN'
```

(The 'ID - 16' select merely ensures that the newest 16 backup files are always excluded from the delete in any case, as an additional safety measure).

The result set is a single-column list of filenames. I stored this in the Object variable **varDeleteFileList** that I'd created earlier using a Recordset Destination control with the 'VariableName' Custom Property set to varDeleteFileList as shown above.

Deleting the files

After this last step I now had a variable containing the list of files that could be safely deleted. To actually delete these files I added another Foreach Loop Container to my package.



The Foreach Loop container allows execution iteration over an enumeration.

Foreach Loop Editor

Enumerator	Foreach ADO Enumerator
Expressions	

Enumerator
Specifies the enumerator type.

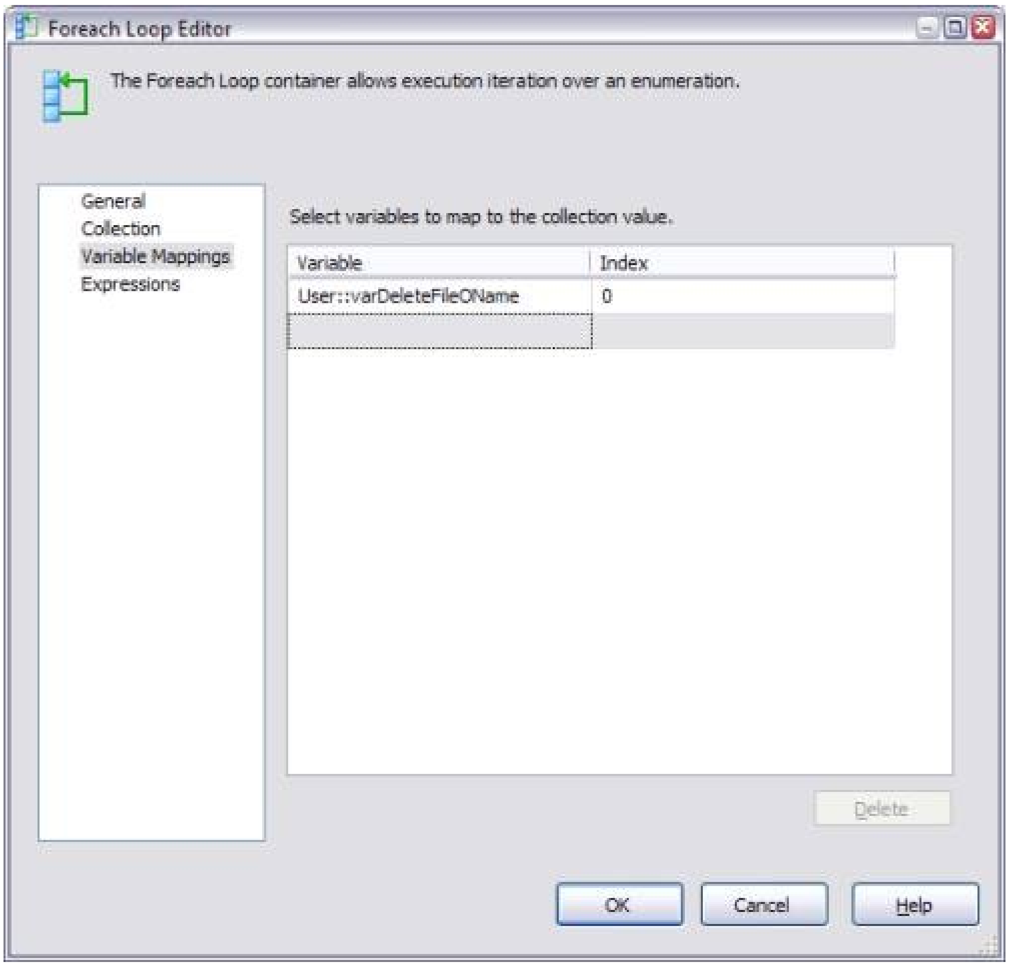
Enumerator configuration

ADO object source variable: User::varDeleteFileList

Enumeration mode

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

This time I selected the 'Foreach ADO enumerator' as the enumerator type. I selected the **varDeleteFileList** variable from the ADO object source variable list and added a mapping to the **varDeleteFileOName** variable on the Variable Mappings tab. This is an Object-typed variable, allowing me to store a recordset. This recordset is then used to delete the files one-by-one as follows.



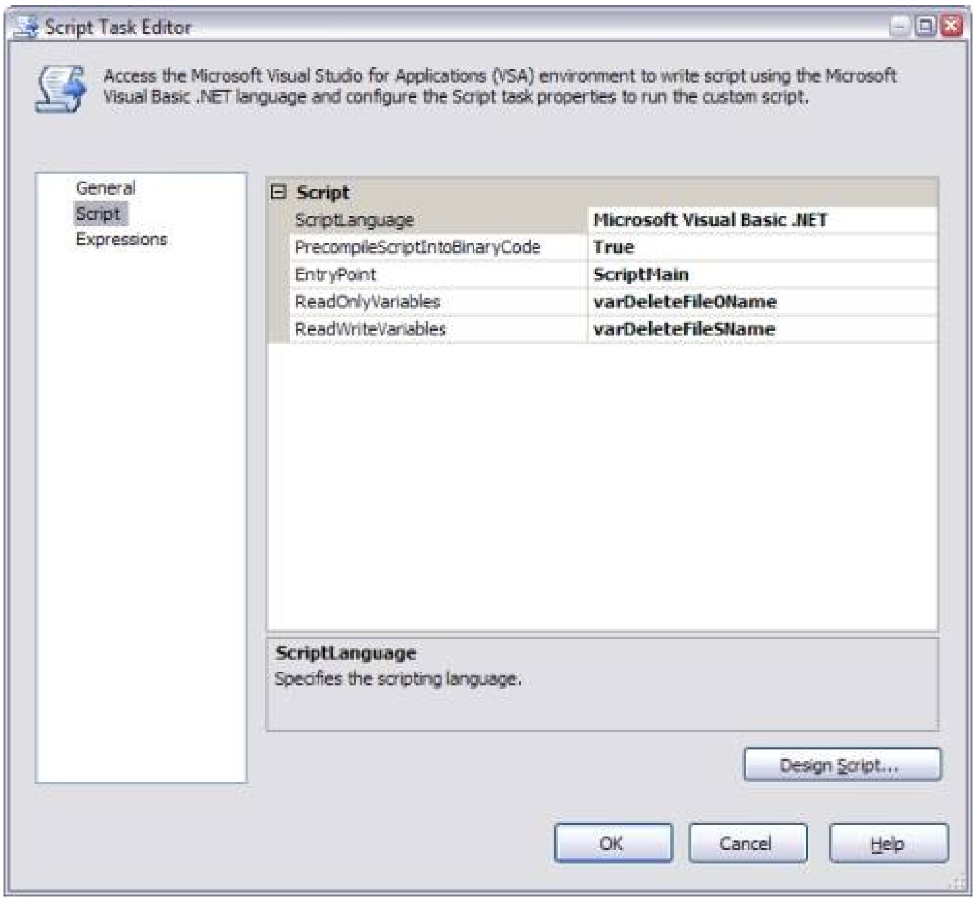
The Foreach Loop container allows execution iteration over an enumeration.

Select variables to map to the collection value.

Variable	Index
User::varDeleteFileOName	0

I added a Script task in the container. This was necessary in order to convert the value of the **varDeleteFileOName** variable from an Object to a string that could be used in the File System Task step (described below).

The Script Task was defined as follows:



Script Task Editor

Access the Microsoft Visual Studio for Applications (VSA) environment to write script using the Microsoft Visual Basic .NET language and configure the Script task properties to run the custom script.

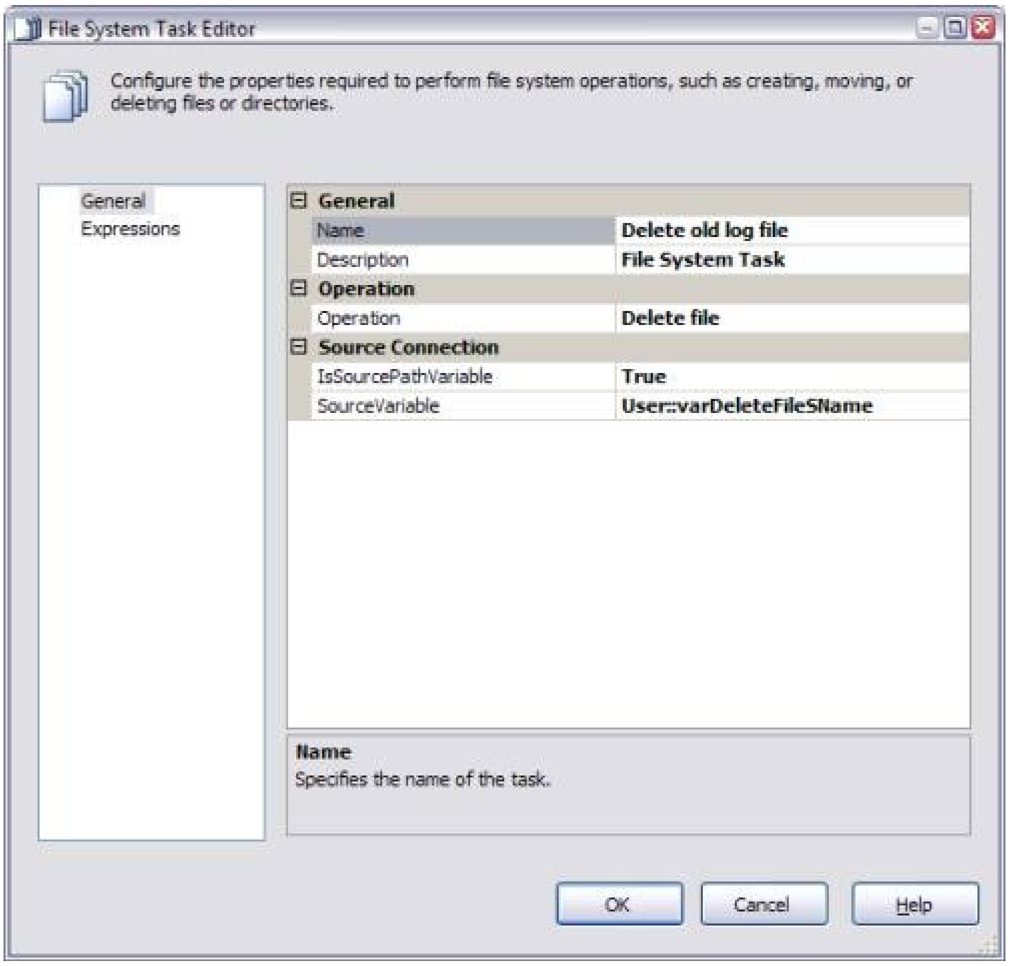
Script	
ScriptLanguage	Microsoft Visual Basic .NET
PrecompileScriptIntoBinaryCode	True
EntryPoint	ScriptMain
ReadOnlyVariables	varDeleteFileOName
ReadWriteVariables	varDeleteFileSName

ScriptLanguage
Specifies the scripting language.

The script code is short and simple:

```
Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Runtime

Public Class ScriptMain

Public Sub Main()

Dim varStringFileName As String

varStringFileName =
Dts.Variables("varDeleteFileOName").Value.ToString

Dts.Variables("varDeleteFileSName").Value =
"\\SERVERNAME\TLog_Backups\" + varStringFileName

Dts.TaskResult = Dts.Results.Success

End Sub

End Class
```

The path listed in the code is, of course, a hard-coded reference to the folder where the files to be deleted are stored but could also be populated by a variable if you so choose. The **varDeleteFileSName** variable is populated with a string-valued path and file name and the Script Task is complete.


File System Task Editor
Configure the properties required to perform file system operations, such as creating, moving, or deleting files or directories.
General
Expressions
OK
Cancel
Help

General	
Name	Delete old log file
Description	File System Task
Operation	
Operation	Delete file
Source Connection	
IsSourcePathVariable	True
SourceVariable	User::varDeleteFileSName

Name
Specifies the name of the task.

The last step in the package was to add a File System Task inside the second Foreach Loop container. On the 'General' tab I selected 'Delete file' in the Operation drop-down, 'True' for IsSourcePathVariable and **varDeleteFileSName** as the SourceVariable.

And that's it! Running the package deletes all backup files that haven't been restored except the newest sixteen. I deployed the package to the production SSIS server and scheduled it with a regular SQL job to run hourly.

This is a fairly simple example that doesn't include much error-trapping, logging and so on, but I hope that it at least illustrates the power and flexibility of the Foreach Loop Container in SQL Server Integration Services.

Paul Clancy
360Data
http://www.360data.nl

Links and References

SQL Books Online – SSIS Basics Tutorial
http://msdn2.microsoft.com/en-us/library/ms169917.aspx

Jamie Thomson's SSIS Blog
http://blogs.conchango.com/jamiethomson/

This article was originally published on SQLServerCentral.com in February 2008.
http://www.sqlservercentral.com/articles/SSIS/61987/

SQL Server Central
http://www.sqlservercentral.com/