SQL Server Integration Services Assignment

Group Submissions: As a bit of a "*get accustomed to working together*" nudge, this assignment is to be a joint submission from your BI course project team. **Only one submission per team**.

Assignment Overview

In this assignment, you will extend the Simple ETL package that we created in our first SSIS lab. See *Labs/Intro to SQL Server Data Tools* on the course Blackboard. Recall that that package extracts data from a single flat file, transforms the data using lookup transformations and finally loads the result into a fact table destination. In Problem 1, you will expand the package to take advantage of control flow looping features to extract multiple flat files into a single data flow process. In Problem 2, you will expand the package you created in Problem 1 to take advantage of package configuration options. Finally, in Problem 3 you will expand the original package to take advantage of SSIS error output capabilities.

You are to complete each of the problems 1 through 3. All of the data, etc. required by this assignment have been provide in the assignment ZIP file. Extract the ZIP file to a folder and rename the *GroupX* folder from the restored ZIP file **replacing the X with the number of your group**. Begin your solutions using the SSDT solution (the "dot sln" file) provided in the *SSIS_Assignment* folder restored from the ZIP file.

Each problem begins with either the completed and functional copy of the **Simple ETL.dtsx** package created in the **Intro to Business Intelligence Developer Studio** lab or the package created by the previous problem. The completed **Simple ETL.dtsx** package for the lab has been provided for you in the SSDT solution.

Instructions for creating a working copy of the appropriate package for each problem are described below.

Deliverables

Once you have completed the assignment, you should have a total of 4 packages in the SSIS project – **Simple ETL.dtsx, Problem1.dtsx, Problem2.dtsx,** and **Problem3.dtsx.** Save all files and exit SSDT. Detach the **SSDT_Lab_DW** database from SQL Server. Create a ZIP file of the *GroupX* folder (now renamed with your group number) and upload that ZIP file to the Blackboard **SSIS Assignment** item.

System Requirements for the SSIS Assignment

To complete this assignment:

- Your system must have SQL Server with the **SSDT_Lab_DW** database attached along with the SSDT development environment (I.e., **SQL Server 2012 Developer** edition). To attach the required database, download and extract the SSDT_LAB.ZIP file to a folder. Start **SQL Server Mgt Studio** and attach the **SSDT_Lab_DW** located in the *GroupX\DataSources* folder restored from the ZIP file. **Note**: be sure to attach the database **after** you have changed the *GroupX* folder name to the appropriate group number for your group. Otherwise, you will not be able to rename the *GroupX* folder.

- This assignment also requires sample data. The sample data is provided in the assignment ZIP file together with the database files in the *GroupX\DataSources* folder.

Assignment Notes

1. To re-run any package in this assignment, you must first delete all rows from the **FactCurrencyRate** table in the **SSDT_Lab_DW** database. You can use the following SQL code in a SQL Server Management Studio query.

```
USE SSDT_Lab_DW
GO
DELETE FROM FactCurrencyRate
```

2. You should make sure that the provided **Simple ETL.dtsx** package executes correctly in your environment before beginning Problem 1. **Important**: before you test or copy the **Simple ETL.dtsx** package for the first time, you must edit the **Sample Flat File Source Data** connection and replace the **File Name** by browsing to the location where you restored the *SampleCurrencyData.txt* file. I.e., the *GroupX\DataSources* folder restored from the ZIP file. Be sure to delete all rows from the **FactCurrencyRate** table in the **SSDT_Lab_DW** database after the test.
3. Since all packages in this assignment essentially update the **FactCurrencyRate** table with the same data, you must also delete all rows from the **FactCurrencyRate** table in the **SSDT_Lab_DW** database before running any **Problem** package after having run any previous **Problem** package. For example, after successfully running the **Problem1.dtsx** package and before running the **Problem2.dtsx** package you must first delete all rows from the **FactCurrencyRate** table.

Problem 1: Adding Looping

In the ***Simple ETL*** lab we created a package that extracted data from a single flat file source, transformed the data using Lookup transformations, and finally loaded the data into the **FactCurrencyRate** fact table of the **SSDT_Lab_DW** database.

However, it is rare for an extract, transform, and load (ETL) process to use a single flat file. A typical ETL process would extract data from multiple flat file sources. Extracting data from multiple sources requires an iterative control flow. One of the most useful features of SSIS is the ability to easily add iteration or looping to packages.

SSIS provides two types of containers for looping through packages: the Foreach Loop container and the For Loop container. The Foreach Loop container uses an enumerator to perform the looping, whereas the For Loop typically uses a variable expression. In this problem, you will modify the **Simple ETL** package to take advantage of the Foreach Loop container. You will also set user-defined package variables to enable the lab exercise package to iterate through all the flat files in the folder. In this problem, you will not modify the data flow, only the control flow.

Task 1: Copying the Simple ETL Package

Start by creating a copy of the **Simple ETL.dtsx** package already provide in the SSIS project solution.

To create the Problem1 package (2 pts)

1. If Business Intelligence Development Studio is not already open, double-click the *SSIS_Assignment.sln* file located in the **SSIS_Assignment** folder.
2. In Solution Explorer, right-click **Simple ETL.dtsx**, and then click **Copy**.
3. In Solution Explorer, right-click **SSIS Packages**, and then click **Paste**.
4. In Solution Explorer, right-click the newly added package and select **Rename**. Rename the package to **Problem1.dtsx**. Select **Ok** when asked to rename the package object.
5. In Solution Explorer, double-click the **Problem1** package to open the Package Designer.
6. In the **Problem1** package properties, click the box for the **ID** property, and then in the list, click **<Generate New ID>**. Each SSIS package object must have a unique ID.

Each of the remaining in Problem 1 tasks is to be performed using the **Problem1.dtsx** package.

Task 2: Adding and Configuring the Foreach Loop Container

In this task, you will add the ability to loop through a folder of flat files and apply the same data flow transformations used in **Simple ETL** to each of those flat files. You do this by adding and configuring a Foreach Loop container to the control flow.

The Foreach Loop container that you add must be able to connect to each flat file in the folder. Because all the files in the folder share an identical internal format, the Foreach Loop container can use the same Flat File connection manager to connect to each of these files. The Flat File connection manager that the container will use is the same Flat File connection manager that you created in **Simple ETL** lab.

Currently, the Flat File connection manager from the **Simple ETL** package connects to only one, specific flat file. To iteratively connect to each flat file in the folder, you will have to configure both the Foreach Loop container and the Flat File connection manager as follows:

- **Foreach Loop container.** You will map the enumerated value of the container to a user-defined package variable. The container will then use this user-defined variable to dynamically modify the **ConnectionString** property of the Flat File connection manager and iteratively connect to each flat file in the folder.
- **Flat File connection manager**. You will modify the connection manager originally created in **Simple ETL** package by using a user-defined variable to populate the connection manager's **ConnectionString** property.

The procedures in this task show you how to create and modify the Foreach Loop container to use a user-defined package variable and to add the data flow task to the loop. You will learn how to modify the Flat File connection manager to use a user-defined variable in the next task.

After you have made these modifications to the package, when the package is run, the Foreach Loop Container will iterate through the collection of files in the Sample Data folder. Each time a file is found that matches the criteria, the Foreach Loop Container will populate the user-defined variable with the file name, map the user-defined variable to the **ConnectionString** property of the Sample Currency Data Flat File connection manager, and then run the data flow against that file. Therefore, each iteration of the Foreach Loop the Data Flow task will “consume” a different flat file.

Notice that because SSIS separates control flow from data flow, any looping that you add to the control flow will not require modification to the data flow. Which is nice.

To add a Foreach Loop container (8 pts)

1. In **SSDT**, click the **Control Flow** tab.
2. In the **Toolbox**, expand **Control Flow Items**, and then drag a **Foreach Loop Container** onto the design surface of the **Control Flow** tab.
3. Right-click the newly added **Foreach Loop Container** and select **Edit**.
4. In the **Foreach Loop Editor** dialog box, on the **General** page, for **Name**, enter **Foreach File in Folder**. Click **OK**.

To configure the enumerator for the Foreach Loop container (10 pts)

1. Double-click Foreach File in Folder to reopen the **Foreach Loop Editor**.
2. Click **Collection**.
3. On the **Collection** page, select **Foreach File Enumerator**.
4. In the **Enumerator configuration** group, click **Browse**.
5. In the **Browse for Folder** dialog box, locate the sample data folder that contains the lab exercise sample data. The sample data is located in the GroupX*DataSources* folder
6. In the **Files** box, type **Currency_*.txt**. If you are familiar with the command line concept of pattern matching or "globbing" this construct will look familiar.

To map the enumerator to a user-defined variable (10 pts)

1. Click **Variable Mappings**.
2. On the **Variable Mappings** page, in the **Variable** column, click the empty cell and select **<New Variable…>**.
3. In the **Add Variable** dialog box, for **Name**, type **varFileName**. Note that SSIS variable names are case sensitive.
4. Click **OK**.
5. Click **OK** again to exit the **Foreach Loop Editor** dialog box.

To add the data flow task to the loop (2 pts)

- Drag the **Extract Sample Currency Data** data flow task onto the Foreach Loop container now renamed **Foreach File in Folder**.

Task 3: Modifying the Flat File Connection Manager

In this task, you will modify the **Sample Flat File Source Data** Flat File connection manager. When originally created, the Flat File connection manager was configured to statically load a single file. To enable the Flat File connection manager to iteratively load files, you must modify the **ConnectionString** property of the connection manager to accept the user-defined variable `User:varFileName`, which, at run time, will contain the path of the file to be loaded.

By modifying the connection manager to use the value of the user-defined variable, `User::varFileName`, to populate the **ConnectionString** property of the connection manager, the connection manager will be able to connect to different flat files. At run time, each iteration of the Foreach Loop container will dynamically update the `User::varFileName` variable. Updating the variable, in turn, causes the connection manager to connect to a different flat file, and the data flow task to process a different set of data.

To configure the Flat File connection manager to use a variable for the connection string (10 pts)

1. In the **Connection Managers** pane, right-click **Sample Flat File Source Data**, and select **Properties**.
2. In the Properties window, for **Expressions**, click in the empty cell, and then click the ellipsis button **(…)**.
3. In the **Properties Expressions Editor** dialog box, in the **Property** column, type or select **ConnectionString**.
4. In the **Expression** column, click the ellipsis button **(…)** to open the **Expression Builder** dialog box.
5. In the **Expression Builder** dialog box, expand the **Variables** node.
6. Drag the variable, **User::varFileName**, into the **Expression** box.
7. Click **OK** to close the **Expression Builder** dialog box.
8. Click **OK** again to close the **Property Expressions Editor** dialog box.

Task 4: Testing the Problem1 Package

With the Foreach Loop container and the Flat File connection manager now configured, the Problem1 package can iterate through the collection of flat files in the *GroupX\DataSources* folder. Each time a file name is found that matches the specified file name criteria, the Foreach Loop container populates the user-defined variable with the file name. This variable, in turn, updates the **ConnectionString** property of the Flat File connection manager, and a connection to the new flat file is made. The Foreach Loop container then runs the unmodified data flow task against the data in the new flat file before connecting to the next file in the folder.

Use the following procedure to test the new looping functionality that you have added to your package.

To test the Problem1 package (2 pts)

1. On **Debug** menu, click **Start Debugging**.

   The package will run. You can verify the status of each loop in the Output window, or by clicking on the **Progress** tab. For example, you can see that 1097 lines were added to the destination table from the file Currency_VEB.txt.

2. After the package has completed, on the **Debug** menu, click **Stop Debugging**.

3. Once you have completed this problem, be sure to delete all rows from the **FactCurrencyRate** table before proceeding to next problem.

Problem 2: Adding Package Configurations

Package configurations let you set run-time properties and variables from outside of the development environment. Configurations allow you to develop packages that are flexible and easy to both deploy and distribute. SSIS offers the following configuration types:

- XML configuration file
- Environment variable
- Registry entry
- Parent package variable
- SQL Server table

In this problem, you will modify the **Problem1.dtsx** package to take advantage of package configurations. Using the Package Configuration Wizard, you will create an XML configuration that updates the **Directory** property of the Foreach Loop container by using a package-level variable mapped to the Directory property. Once you have created the configuration file, you will modify the value of the variable from outside of the development environment and point the modified property to a new sample data folder. When you run the package again, the configuration file populates the value of the variable, and the variable in turn updates the **Directory** property. As a result, the package iterates through the files in the new data folder, rather than iterating through the files in the original folder path that was hard-coded in the package.

Task 1: Copying the Problem1 Package

In this task, you will create a copy of the Problem1.dtsx package that you created in Problem1. You will use this new copy throughout the remainder of Problem2.

To create the Problem2 package (2 pts)

1. Follow the steps from Task 1 in Problem 1 except start by copying the **Problem1.dtsx** package to create the **Problem2.dtsx** package.

Task 2: Enabling and Configuring Package Configurations

In this task, you will enable package configurations using the Package Configuration Wizard. You will use this wizard to generate an XML configuration file that contains configuration settings for the **Directory** property of the Foreach Loop container. The value of the Directory property is supplied by a new package-level variable that you can update at run time. Additionally, you will populate a new sample data folder to use during testing.

To create a new package-level variable mapped to the Directory property (8 pts)

1. Click the background area of the **Control Flow** tab in SSIS Designer. This sets the scope for the variable you will create to the package.
2. On the SSIS menu, select **Variables**.
3. In the **Variables** window, click the Add Variable icon.
4. In the **Name** box, type **varFolderName**.
5. Verify that the **Scope** box shows the name of the package, Problem2.
6. Set the value of the **Data Type** box of the **varFolderName** variable to **String**.
7. Return to the **Control Flow** tab and double-click the **Foreach File in Folder** container.
8. On the **Collection** page of the **Foreach Loop Editor**, click **Expressions**, and then click the ellipsis button **(…)**.
9. In the **Property Expressions Editor**, click in the **Property** list, and select **Directory**.
10. In the **Expression** box, click the ellipsis button **(…)**.
11. In the **Expression Builder**, expand the Variables folder, and drag the variable **User::varFolderName** to the **Expression** box.
12. Click **OK** to exit the **Expression Builder**.
13. Click **OK** to exit the **Property Expressions Editor**.

To enable package configurations (8 pts)

1. Right-click the background of the **Control Flow** tab in SSIS Designer.
2. On the menu, click **Properties**. In the properties window, locate and select the Configurations entry. Click on the ellipse (three dots) on the right-hand-side.
3. In the **Package Configurations Organizer** dialog box, select **Enable Package Configurations**, and then click **Add**.
4. On the welcome page of the Package Configuration Wizard, click **Next**.
5. On the **Select Configuration Type** page, verify that the **Configuration type** is set to **XML configuration file**.
6. On the **Select Configuration Type** page, click **Browse**.
7. By default, the **Select Configuration File Location** dialog box will open to the project folder.
8. In the **Select Configuration File Location** dialog box, for **File name** type **SSIS_Assignment**, and then click **Save**.
9. On the **Select Configuration Type** page, click **Next.**
10. On the **Select Properties to Export** page, in the **Objects** pane, expand **Variables**, expand **varFolderName**, expand **Properties**, and then select **Value**.

11. On the **Select Properties to Export** page, click **Next**.
12. On the **Completing the Wizard** page, type a configuration name for the configuration, such as **SSIS Lab exercise Directory configuration**. This is the configuration name that is displayed in the **Package Configuration Organizer** dialog box.
13. Click **Finish**.
14. Click **Close**.
15. The wizard creates a configuration file named **SSIS_Assignment**.**dtsConfig** that contains configuration settings for the v**alue** of the variable that in turn sets the **Directory** property of the enumerator. A configuration file typically contains complex information about the package properties, but for this assignment the only information in the file should be **[User::varFolderName].Properties[Value]**.

To create and populate a new sample data folder (2 pts)

1. In Windows Explorer, create a new folder named **New Sample Data** in the *GroupX\DataSources* folder.
2. Open the *GroupX\DataSources* folder and then copy any three of the *Currency_*.txt* sample files from the folder.
3. In the **New Sample Data** folder, paste the copied files.

Task 3: Modifying the Directory Property Configuration Value

In this task, you will modify the configuration setting, stored in the **SSIS_Assignment**.**dtsConfig** file, for the **Value** property of the package-level variable `User::varFolderName`. The variable updates the **Directory** property of the Foreach Loop container. The modified value will point to the **New Sample Data** folder that you created in the previous task. After you modify the configuration setting and run the package, the **Directory** property will be updated by the variable, using the value populated from the configuration file instead of the directory value originally configured in the package.

To modify the configuration setting of the Directory property (8 pts)

1. In Notepad or any other text editor, locate and open the **SSIS_Assignment**.**dtsConfig** configuration file that you created by using the Package Configuration Wizard in the previous task.
2. Change the value of the **ConfiguredValue** element to match the path of the **New Sample Data** folder that you created in the previous task. Do not surround the path in quotes. If the **New Sample Data** folder is at the root level of your thumb-drive (for example, J:\), the updated XML should be similar (but not exact) to the following sample:

```
<?xml
version="1.0"?><DTSConfiguration><DTSConfigurationHeading><DTSConfigur
ationFileInfo GeneratedBy="Domain\UserName"
GeneratedFromPackageName="Problem2" GeneratedFromPackageID="{99396D72-
2F8D-4A37-8362-96346AD53334}" GeneratedDate="11/12/2005 12:46:13
PM"/></DTSConfigurationHeading><Configuration
```

```
ConfiguredType="Property"
Path="\Package.Variables[User::varFolderName].Properties[Value]"
ValueType="String"><ConfiguredValue>J:GroupX\DataSources\New Sample
Data</ConfiguredValue></Configuration></DTSConfiguration>
```

The heading information, **GeneratedBy**, **GeneratedFromPackageID**, and **GeneratedDate** will be different in your file, of course. The element to note is the **Configuration** element. The **Value** property of the variable, `User::varFolderName`, now contains *J:\GroupX\DataSources\New Sample Data*.

3. Save the change, and then close the text editor.

Task 4: Testing the Problem2 Lab exercise Package

At run time, your package will obtain the value for the **Directory** property from a variable updated at run time, rather than using the original directory name that you specified when you created the package. The value of the variable is populated by the **SSIS_Assignment**.**dtsConfig** file.

To verify that the package updates the Directory property with the new value during run time, simply execute the package. Because only three sample data files were copied to the new directory, the data flow will run only three times, rather than iterate through the 14 files in the original folder.

To test the Problem2 package (2 pts)

1. On the **Debug** menu, click **Start Debugging**.
2. After the package has completed running, on the **Debug** menu, and then click **Stop Debugging**.
3. Once you have completed this problem, be sure to delete all rows from the **FactCurrencyRate** table before running the next problem.

Problem 3: Adding Error Flow Redirection

To handle errors that may occur in the transformation process, SSIS gives you the ability to decide on a per component and per column basis how to handle data that cannot be transformed. You can choose to ignore a failure in certain columns, redirect the entire failed row, or just fail the component. By default, all components in SSIS are configured to fail when errors occur. Failing a component, in turn, causes the package to fail and all subsequent processing to stop. Which may or may not be what you want.

Instead of letting failures stop package execution, it is good practice to configure and handle potential processing errors as they occur within the transformation. While you might choose to ignore failures to ensure your package runs successfully, it is often better to redirect the failed row to another processing path where the data and the error can be persisted, examined and reprocessed at a later time.

In this problem, you will create a copy of the package that you developed for Problem 1. Working with this new package, you will create a corrupted version of one of the sample data files. The corrupted file will force a processing error to occur when you run the package.

To handle the error data, you will add and configure a Flat File destination that will write any rows that fail to locate a lookup value in the Lookup Currency Key transformation to a file.

Before the error data is written to the file, you will include a Script component that uses script to get error descriptions. You will then reconfigure the Lookup Currency Key transformation to redirect any data that could not be processed to the Script transformation.

Task 1: Copying the Problem1 Package

To create the Problem3 package (2 pts)

1. Follow the steps from Task 1 in Problem 1 except start by copying the **Problem1.dtsx** package to create the **Problem3.dtsx** package.

Task 2: Creating a Corrupted File

In order to demonstrate the configuration and handling of transformation errors, you will have to create a sample flat file that when processed causes a component to fail.

In this task, you will create a copy of an existing sample flat file. You will then open the file in Notepad and edit the **CurrencyID** column to ensure that it will fail to produce a match during the transformations lookup. When the new file is processed, the lookup failure will cause the Currency Key Lookup transformation to fail and therefore fail the rest of the package. After you have created the corrupted sample file, you will run the package to view the package failure.

To create a corrupted sample flat file (2 pts)

1. In Notepad or any other text editor, open the *Currency_VEB.txt* file located in the *GroupX\DataSources* folder.
2. Use a text editor's (i.e., Notepad) to find and replace feature to find all instances of **VEB** and replace them with **BAD**.
3. In the same folder as the other sample data files, save the modified file as *Currency_BAD.txt*.
4. Make sure that *Currency_BAD.txt* is saved to the *GroupX\DataSources* folder.
5. Close your text editor.

To verify that an error will occur during run time

1. On the **Debug** menu, click **Start Debugging**.

   On the third iteration of the data flow, the Lookup Currency Key transformation tries to process the Currency_BAD.txt file, and the transformation will fail. The failure of the transformation will cause the whole package to fail.
2. On the **Debug** menu, click **Stop Debugging**.
3. On the design surface, click the **Execution Results** tab.
4. Browse through the log and verify that the following unhandled error occurred:

   ```
   [Lookup Currency Key[30]] Error: Row yielded no match during lookup.
   ```

   Note, the number 30 is the ID of the component. This value is assigned when you build the data flow, and the value in your package will likely be different.

4. Delete all rows from the **FactCurrencyRate** table.

Task 3: Adding Error Flow Redirection

As demonstrated in the previous task, the Lookup Currency Key transformation cannot generate a match when the transformation tries to process the corrupted sample flat file, which produced an error. Because the transformation uses the default settings for error output, any error causes the transformation to fail. When the transformation fails, the rest of the package also fails.

Instead of permitting the transformation to fail, you can configure the component to redirect the failed row to another processing path by using the error output. Use of a separate error processing path lets you do a number of things. For instance, you might try to clean the data and then reprocess the failed row. Or, you might save the failed row along with additional error information for later verification and reprocessing.

In this task, you will configure the Lookup Currency Key transformation to redirect any rows that fail to the error output. In the error branch of the data flow, these rows will be written to a file.

By default the two extra columns in an SSIS error output, **ErrorCode** and **ErrorColumn**, contain only numeric codes that represent an error number, and the ID of the column in which the error occurred. These numeric values may be of limited use without the corresponding error description.

To enhance the usefulness of the error output, before the package writes the failed rows to the file, you will use a Script component to access the SSIS API and get a description of the error.

To configure an error output (10 pts)

1. In the **Toolbox**, expand **Data Flow Transformations**, and then drag **Script Component** onto the design surfaceof the **Data Flow** tab. Place **Script** to the right of the **Lookup Currency Key** transformation.
2. In the **Select Script Component Type** dialog box, click **Transformation**, and click **OK**.
3. Click the **Lookup Currency Key** transformation and then drag the red arrow onto the newly added **Script** transformation to connect the two components.

   The red arrow represents the error output of the **Lookup Currency Key** transformation. By using the red arrow to connect the transformation to the Script component, you can redirect any processing errors to the Script component, which then processes the errors and sends them to the destination.
4. In the **Configure Error Output** dialog box, in the **Error** column, select **Redirect row**, and then click **OK**.
5. On the **Data Flow** design surface, click **Script Component** in the newly added **Script Component**, and change the name to **Get Error Description**.
6. Double-click the **Get Error Description** transformation.
7. In the **Script Transformation Editor** dialog box, on the **Input Columns** page, select the **ErrorCode** column.
8. On the **Inputs and Outputs** page, expand **Output 0**, click **Output Columns**, and then click **Add Column**.

9. In the **Name** property, type **ErrorDescription** and set the **DataType** property to **Unicode [DT_WSTR]**.
10. On the **Script** page, verify that the **LocaleID** property is set to **English (United States)** and that **ScriptLanguage** is set to **Microsoft Visual Basic 2010**. If **Microsoft Visual C# 2010** is the only available scripting language on your system, use the code from step 11.B. Otherwise, use step 11.A.
11. **A**. Click the **Edit Script…** button to open Microsoft Visual Studio Tools for Applications (VSTA). In the **Input0_ProcessInputRow** method, type or paste the following Visual Basic code.

```
  Row.ErrorDescription =
    Me.ComponentMetaData.GetErrorDescription(Row.ErrorCode)
```

The completed subroutine will look like the following code.

```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)


  Row.ErrorDescription =
    Me.ComponentMetaData.GetErrorDescription(Row.ErrorCode)


End Sub
```

B. Click **Edit Script** to open Microsoft Visual Studio Tools for Applications (VSTA). In the **Input0_ProcessInputRow** method, type or paste the following Visual C# code.

```
  Row.ErrorDescription =
    this.ComponentMetaData.GetErrorDescription(Row.ErrorCode);
```

The completed subroutine will look like the following code.

```
public override void Input0_ProcessInputRow(Input0Buffer Row)
    {
        Row.ErrorDescription =
           this.ComponentMetaData.GetErrorDescription(Row.ErrorCode);
    }
```

12. Build the script by selecting **Build** from the VSTA menu to save your changes. Then close VSTA by selecting **Exit** from the VSTA **File** menu.
13. Click **OK** to close the **Script Transformation Editor** dialog box.

Task 4: Adding a Flat File Destination

The error output of the Lookup Currency Key transformation redirects to the Script transformation any data rows that failed the lookup operation. To enhance information about the errors that occurred, the Script transformation runs a script that gets the description of errors.

In this task, you will save all this information about the failed rows to a delimited file for later processing. To save the failed rows, you must add and configure a Flat File connection manager for the text file that will contain the error data and a Flat File destination. By setting properties on the Flat File connection manager that the Flat File destination uses, you can specify how the Flat File destination formats and writes the text file.

To add and configure a Flat File destination (10 pts)

1. Click the **Data Flow** tab.
2. In the **Toolbox**, expand **Data Flow Destinations**, and drag **Flat File Destination** onto the data flow design surface. Put the **Flat File Destination** directly underneath the **Get Error Description** transformation.
3. Click the **Get Error Description** transformation, and then drag the green arrow onto the new **Flat File Destination**.
4. On the **Data Flow** design surface, click **Flat File Destination** in the newly added **Flat File Destination** transformation, and change the name to **Failed Rows**.
5. Right-click the **Failed Rows** transformation, click **Edit**, and then in the **Flat File Destination Editor**, click **New**.
6. In the **Flat File Format** dialog box, verify that **Delimited** is selected, and then click **OK**.
7. In the **Flat File Connection Manager Editor**, in the **Connection Manager Name** box type **Error Data**.
8. In the **Flat File Connection Manager Editor** dialog box, click **Browse**, and locate the folder in which to store the file. Store the file in the *GroupX\DataSources* folder.
9. In the **Open** dialog box, for **File name**, type **ErrorOutput.txt**, and then click **Open**.
10. In the **Flat File Connection Manager Editor** dialog box, verify that the **Locale** box contains English (United States) and **Code page** contains 1252 (ANSI -Latin I).
11. In the options pane, click **Columns**.

    Notice that, in addition to the columns from the source data file, three new columns are present: ErrorCode, ErrorColumn, and ErrorDescription. These columns are generated by the error output of the Lookup Currency Key transformation and by the script in the Get Error Description transformation, and can be used to troubleshoot the cause of the failed row.
12. Click **OK**.
13. In the **Flat File Destination Editor**, clear the **Overwrite data in the file** check box.

    Clearing this check box persists the errors over multiple package executions.
14. In the **Flat File Destination Editor**, click **Mappings** to verify that all the columns are correct. Optionally, you can rename the columns in the destination.
15. Click **OK**.

Task 5: Testing the Problem3 Package

At run time, the corrupted file, *Currency_BAD.txt*, will fail to generate a match within the Currency Key Lookup transformation. Because the error output of Currency Key Lookup has now been configured to redirect failed rows to the new Failed Rows destination, the component does not fail, and the package runs successfully. All failed error rows are written to *ErrorOutput.txt*.

In this task, you will test the revised error output configuration by running the package. Upon successful package execution, you will then view the contents of the *ErrorOutput.txt* file. If you do not want to accumulate error rows in the ErrorOutput.txt file, you should manually delete the file content between package runs.

To test the Problem3 package verify the contents of the ErrorOutput.txt file (2 pts)

1. On the **Debug** menu, click **Start Debugging**.
2. After the package has completed running, on the **Debug** menu, and then click **Stop Debugging**.
3. Once you have completed this problem, do **NOT** delete the rows from the **FactCurrencyRate** table before submitting your assignment.
4. In Notepad or any other text editor, open the *ErrorOutput.txt* file. The default column order is: **AverageRate**, **CurrencyID**, **CurrencyDate**, **EndOfDateRate**, **ErrorCode**, **ErrorColumn**, **ErrorDescription**.

   Notice that all the rows in the file contain the unmatched CurrencyID value of BAD, the ErrorCode value of -1071607778, the ErrorColumn value of 0, and the ErrorDescription value "*Row yielded no match during lookup*". The value of the ErrorColumn is set to 0 because the error is not column specific. It is the lookup operation that failed.

Wrap-Up

To complete the assignment save all SSDT project files and exit the program. To submit your group's assignment, follow the instructions given in the **Deliverables** section at the top of this document.