ETL Application Design and Implementation

Prepared For

Payment History Loading Process

Prepared By

Ajay kumar Ramineni
Student ID: 1743420
University: Fairleigh Dickinson University

Table of Contents

I. Overview

The ETL framework provides a shell for structured and standardized data extraction, transformation, and loading processes. The ETL framework is designed to accommodate multiple types of ETL processes by requiring each process to include the same types of ETL process steps while allowing the internal details of each step to be tailored to individual system needs. Thus, ETL processes for heterogeneous data sources and destinations can be defined and hosted by the same ETL framework. The ETL framework is designed to provide the functionality required by an automated ETL process in order to populate and manage a business intelligence system. However, the flexibility of the ETL framework enables the use of the framework to perform much simpler data movement processes that may or may not include steps such as data transformation.

II. Document Conventions

The following abbreviations appear in this document.

Abbreviations	
Abbreviation	**Reference**
ETL	Extraction, Transformation, and Loading
GUID	Globally Unique Identifier
OLAP	Online Analytical Processing
QA	Quality Assurance
SQL	Structured Query Language
SSIS	SQL Server Integration Services

III. ETL Framework Components

The ETL framework consists of databases, database tables, SSIS packages, an XML file, and an environment variable.

A. Databases

The following databases are used by the ETL framework to define and execute ETL processes.

1. Configuration

The ETL framework configuration database is used to define ETL processes, control ETL process execution, take measurements of data processed, and to log progress and results. The configuration database is shared by multiple ETL processes. If more than one ETL framework configuration database is to be on a server, SSIS packages using separate configuration databases must use separate environment variables that point to the ETL framework configuration file.

2. Staging

There is a staging database for each loading destination named in the ETL instance table. The tables in the staging database are structured similarly to the tables in the loading destination. Unlike a loading destination that is dimensional model-based tables, the staging tables do not include a surrogate key column. Because the number of staging databases depends upon the number of loading destinations, multiple ETL instances may utilize the same staging database.

3. Loading

If a database, the loading destination is likely to be dimensional model-based tables. However, because the ETL framework is designed to perform heterogeneous ETL processes, the loading destination can be of any type supported by the OLE-DB destination objects in SSIS data flow tasks.

If the loading destination is dimension model-based tables, the loading database will contain a stored procedure for each dimension table that initializes the table with a record to represent the unknown member in OLAP dimensions.

B. Configuration and Control Tables

The following tables are used by the ETL framework in the configuration database to define and control ETL processes.

Configuration and Control Tables	
Table	**Description**
sysdtslog	The table used by SSIS for logging package events.
SSIS Configurations	All package level Dynamic configurations are stored in this table we can manage environment to environment
SSIS_Event_Log	This table used by SSIS Event Handling mechanism

1. sysdtslog

The table sysdtslog is created and used by SSIS when an SQL Server connection manager is specified for package logging. The sysdtslog table does not have to reside on the same server as the ETL framework configuration database. The ETL framework's logging tables contain most of the same information in the sysdtslog table, making joins possible between the tables. Also, the presence of sysdtslog data in the ETL framework logging tables make the ETL framework’s log purge schedule independent of the purging schedule for sysdtslog.

2. [SSIS Configurations]

[SSIS Configurations] is used to store definitions of ETL processes to perform within the ETL framework. Each ETL instance specifies configuration, import (extracted data landing), staging, and loading database connection information. Each ETL instance also designates SSIS package names to execute at each stage of the ETL process. ETL process definitions for multiple data sources can be used to load data into separate destinations. Another use of multiple instances is to keep both QA and production instance records in the instance table when deploying the framework to production so that a small QA subset of the data can be used with the framework to test the production environment before loading a very large production data set.

2 [Note (Tips) * Important for interview]

Most organizations start out creating SSIS package one by one until they have dozens, hundreds, or even thousands of packages. This can be a nightmare to maintain. You can save yourself a lot of work by deciding upfront how to configure your packages using configuration files or tables. We implemented example of passing information to a package with a configuration table. Then we will go over using configuration tables on multiple packages. Imagine running dozens of packages that point to a server and the server name changes. If you have a configuration table that is feeding this server name to every package you can make a single change to the configuration table and all the packages are updated. This can reduce your maintenance time significantly.

3 [SSIS Event Log]

SSIS event handlers are the simplest means of turning an SSIS script into a reliable system that is auditable, reacts appropriately to error conditions, reports progress and allows instrumentation and monitoring your SSIS packages. They are easy to implement, and provide a great deal of flexibility.

IV. ETL Process for Payment History Loading Process

A. Extract logic

1. Purpose

- XYZ Inc. (Example Company) will be using the Microsoft SSIS ETL tool to load payment history files from different vendors and customers from the FTP or shared network drive.

- The purpose of this specification is to outline the logic to load an ETL file from the payment history from vendor/customer payment file provided by XYZ Inc. interface system.

- A **pro_no** is uniquely identified by invoice number which is the identification number to identify shipment in the system. This pro_no is also used to identify the tax GL Accounting and account related data in the GL Accounting system.

- A Pro_no may be in different states in the accounting system like Fully paid, partially paid, Not paid.

- When the pro_no is Fully Paid then system needs to ignore incoming pro_no as duplicate payment

- When pro_no is partially paid then system needs to add existing paid amount to incoming record from payment history file.

- When the pro_no is Not Paid then system needs to create new record in the accounting table as payment started for that invoice

3. Extract Logic

Read one file at a time from the input folder and apply the business validations on that data ,if any data does not satisfy business logic redirect those records to correct tables after adding corresponding error message to it . Whatever the data does satisfy business validations then we need to move that data to final destination tables.

4. Assumptions

- The payment history file will be provided from by vendors/customers each every day and will contain invoice and paid amounts posted in accounting system for a single processing day. There is not a specific month end file, each day of the month is treated the same.

- The Microsoft SSIS ETL tool will not do any transformation of the Trade data(any additions or subtractions or changing date).

- The “Complete List of Data Elements” includes a complete list of the fields to be extracted from the incoming payment history file(All the Fields).

5. File Format and Naming Conventions

- The extract file name from Microsoft SSIS ETL tool will be “vendor/CustoemrName_datetimestamp”.
- The extract file from vendor/customer for payment data will be in ASCII format, pipe (“~”) delimited. This file will contain either header and no trailer records.
- Unless otherwise indicated in the Data Map section below, all dates will be in the form of YYYYMMDD, SQL large date format in the ETL file.
- Unless otherwise indicated in the Data Map section below, all floating-point numbers will contain an explicit decimal point.

Data Elements from Payment History file to Accounting System.

Field No	**Field Name**	**Table Name**	**Field Description**	**Format**
1	Customer Key	PH_Account	To Identify the invoice is belongs to which customer	INT
2	SCAC	PH_Account	To identify the invoice is belongs to which carrier (Shiper)	Char(10)
3	Pro_no	PH_Account	Identify the invoice	VARCAHR(25)
4	Entry_date	PH_Account	Invoice System entry date	DateTime
5	Ship_Date	PH_Account	Invoice Ship Date	DateTime
6	Billed Amount	PH_Account	Invoice Billed Amount	Numeric(18,2)
7	Paid amount	PH Account	Invoice paid amount	Numeric(18,2)

6. Selection Criteria /Business Logic /Validations

.

- To determine the list of invoice to send to accounting the Microsoft SSIS ETL tool will select all records from the Payment History interface file (Input File).

- Validate Does any record is missing Customer Key Value , If any record missing customer key then redirect to error table after adding corresponding error message to that record.

- Validate Does any record is missing SCAC Value , If any record missing SCAC then redirect to error table after adding corresponding error message to that record.

- Validate Does any record is missing PRO_NO, If any record missing PRO_NO then redirect to error table after adding corresponding error message to that record.

- If incoming invoice exist in the destination table with fully paid then we need to ignore that record (Logic based on pro_no and Billed amount).(if source and destination pro_no and billed amount is same then it is fully paid)

- If incoming invoice exist in the destination table with partially paid then we need to update destination table with incoming billed amount (Logic based on pro_no and Billed amount).(if source billed amount is greater than destination record billed amount)

- If incoming invoice not exists in the destination table then insert that record as not paid or payment started

7. Validation for Accuracy and Completeness

- Needs to implement validation of input file total records vs loaded and eroded record count should be match
- If total no of records in the input file and sum of loaded and error record count is mismatch then we need to set flag for Accuracy and Completeness check failed and someone needs to review the data

8. File Exceptions (Bad Format or File not readable)

- If Microsoft SSIS Tool unable to read input file then that file needs to be redirected to EXCEPTION FOLDER
- If DATA Flow Task failed middle of loading the flat file then that file needs to redirected to EXCEPTION FOLDER

V. Dynamic Configuration Variables and Connection Strings

A Package Variables

1) File Exceptions (Bad Format or File not readable)

B Package Connection Strings

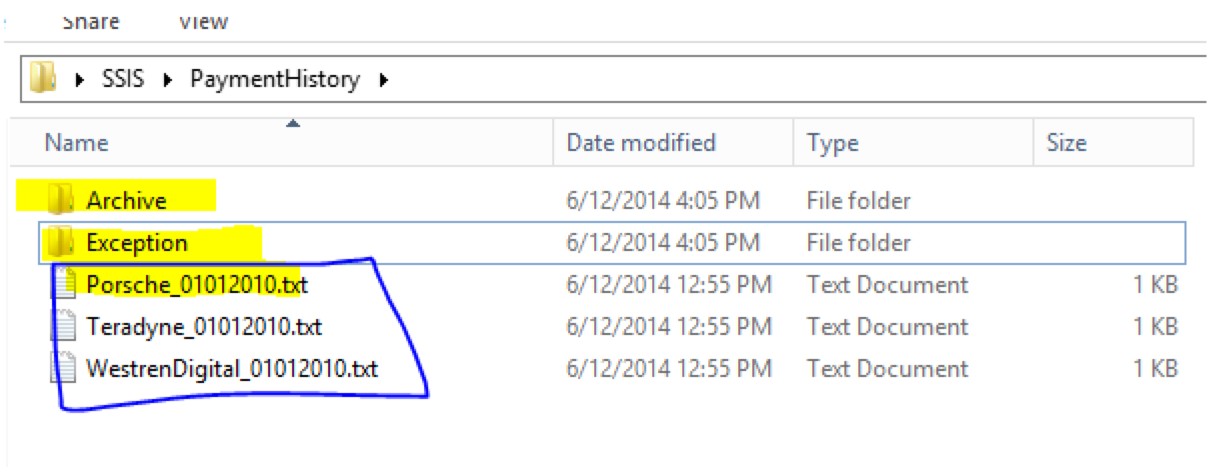
VI. Technical Package Implementation

Create the following tables before starting the process

```
CREATE TABLE PH_Account(
Customer_Key VARCHAR(50),
SCAC VARCHAR(50),
PRO_NO VARCHAR(50),
Billed_amount VARCHAR(50),
Piad_amount VARCHAR(50),
Entry_date VARCHAR(50),
Ship_Date VARCHAR(50),
FileNames NVARCHAR(250)
)
```

```
CREATE TABLE PH_Account_ErrorTable(
Customer_Key VARCHAR(50),
SCAC VARCHAR(50),
PRO_NO VARCHAR(50),
Billed_amount VARCHAR(50),
Piad_amount VARCHAR(50),
Entry_date VARCHAR(50),
Ship_Date VARCHAR(50),
ErrorMsg nvarchar(1000),
FileNames NVARCHAR(250)
)
```

Step 1: Verify source directory for input files

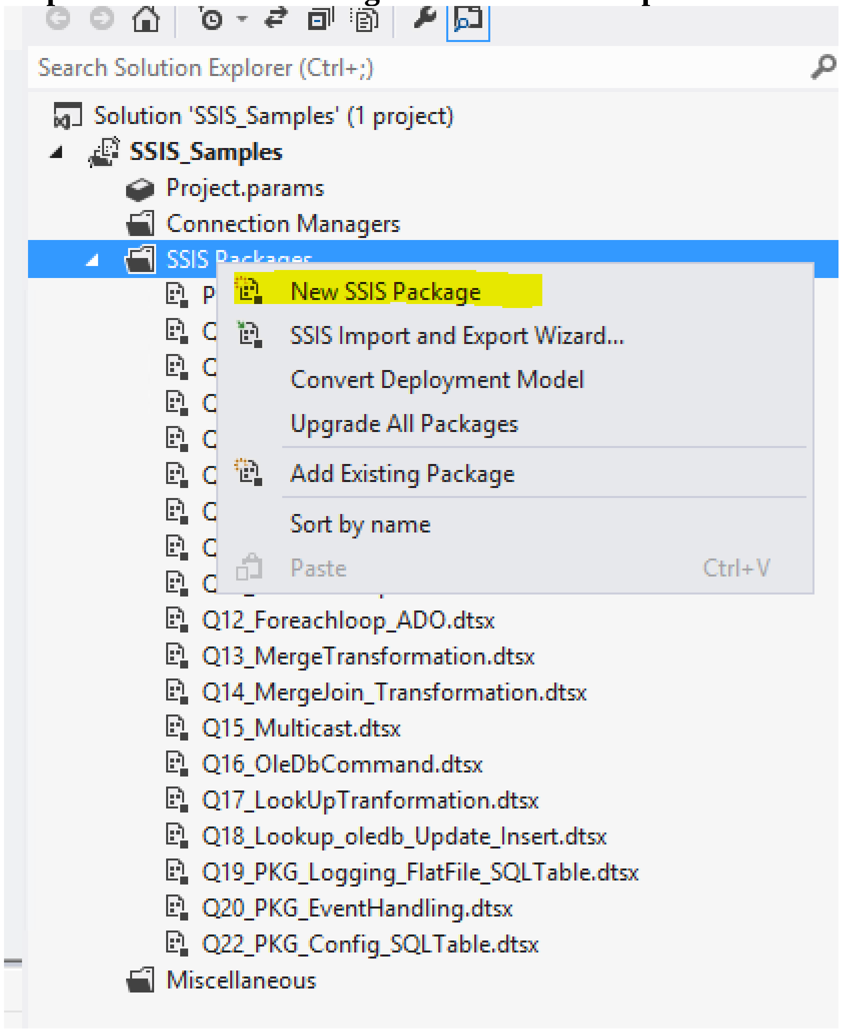


Share View

SSIS ▸ PaymentHistory ▸

Name	Date modified	Type	Size
Archive	6/12/2014 4:05 PM	File folder	
Exception	6/12/2014 4:05 PM	File folder	
Porsche_01012010.txt	6/12/2014 12:55 PM	Text Document	1 KB
Teradyne_01012010.txt	6/12/2014 12:55 PM	Text Document	1 KB
WestrenDigital_01012010.txt	6/12/2014 12:55 PM	Text Document	1 KB

Step 2: Create SSIS Package in the solution Explorer



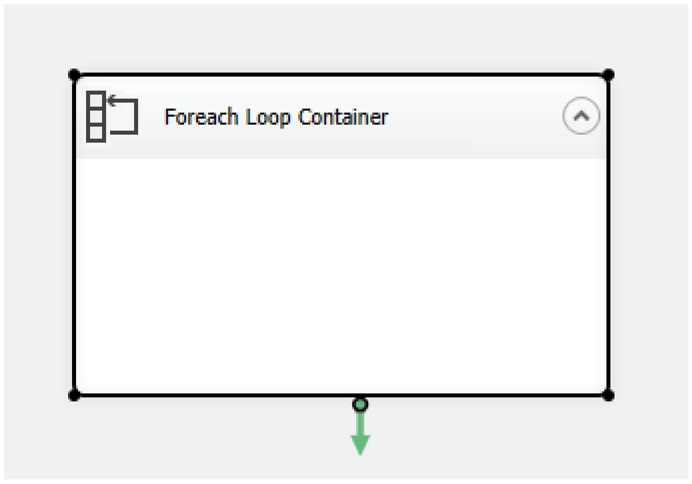
Search Solution Explorer (Ctrl+;)

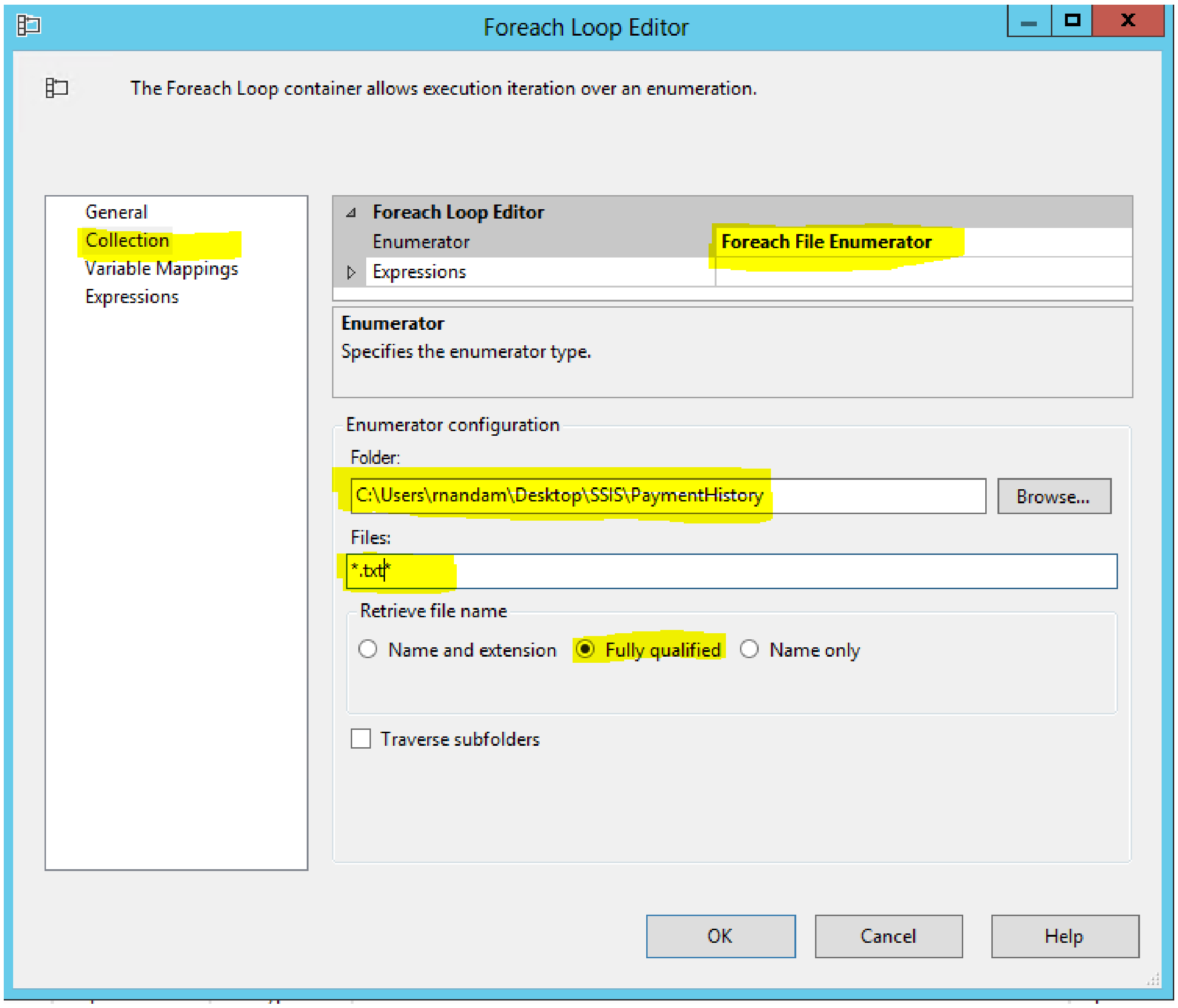
Solution 'SSIS_Samples' (1 project)
SSIS_Samples
Project.params
Connection Managers
SSIS Packages
New SSIS Package
SSIS Import and Export Wizard...
Convert Deployment Model
Upgrade All Packages
Add Existing Package
Sort by name
Paste Ctrl+V
Q12_Foreachloop_ADO.dtsx
Q13_MergeTransformation.dtsx
Q14_MergeJoin_Transformation.dtsx
Q15_Multicast.dtsx
Q16_OleDbCommand.dtsx
Q17_LookUpTranformation.dtsx
Q18_Lookup_oledb_Update_Insert.dtsx
Q19_PKG_Logging_FlatFile_SQLTable.dtsx
Q20_PKG_EventHandling.dtsx
Q22_PKG_Config_SQLTable.dtsx
Miscellaneous

Step 3: Drag and Drop For each loop Container and Config the following

a) Source File Folder (Create the variable to make it dynamic)
b) Select enumeration type (For each file enumerator)
c) Select type of the file

d) Declare the Variable for file name (to make flat file connection dynamic)





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: C:\Users\rnandam\Desktop\SSIS\PaymentHistory

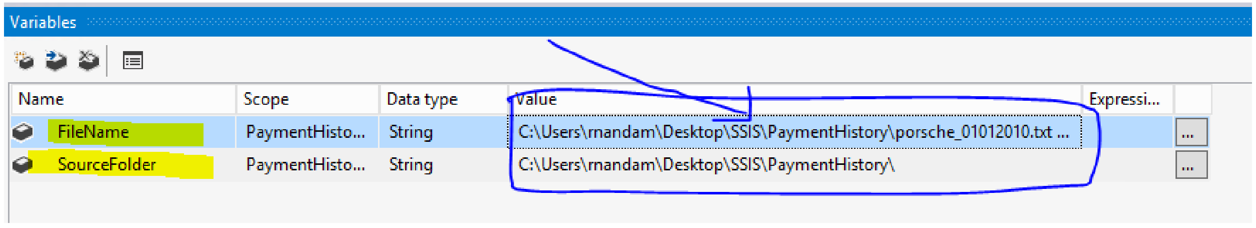
Files: *.txt*

Retrieve file name: Name and extension [✓ Fully qualified] Name only

☐ Traverse subfolders

Step 4: Declare the Variables

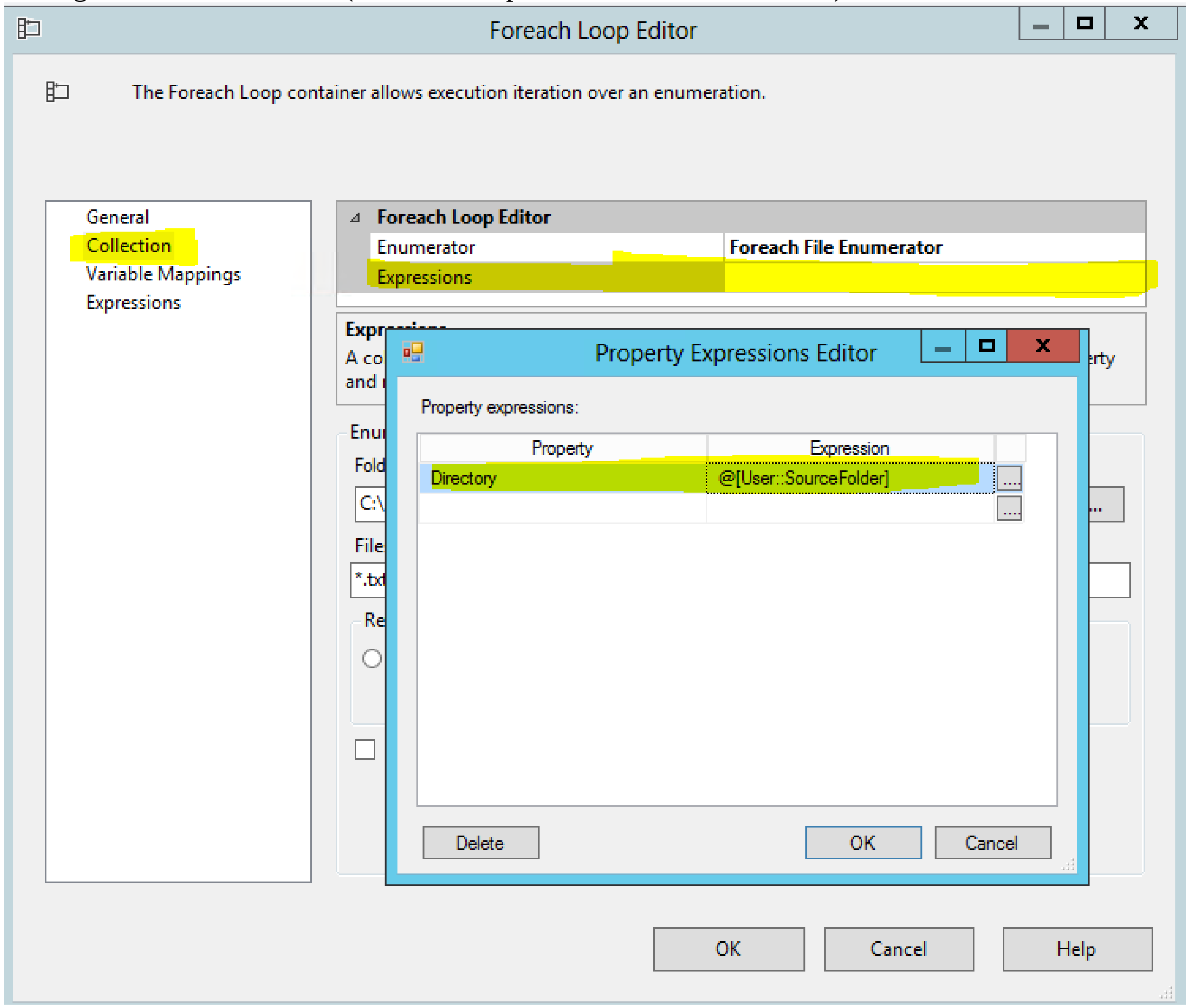
a) Filename (provider default value)
b) SourceFolder (provide the default Value)


Variables

Name	Scope	Data type	Value	Expressi...
FileName	PaymentHisto...	String	C:\Users\rnandam\Desktop\SSIS\PaymentHistory\porsche_01012010.txt ...	
SourceFolder	PaymentHisto...	String	C:\Users\rnandam\Desktop\SSIS\PaymentHistory\	

Step 5: Configure the Variables in the for each loop container

a) Configure for source folder (Click on Expression under Enumerator)


Foreach Loop Editor

The Foreach Loop container allows execution iteration over an enumeration.

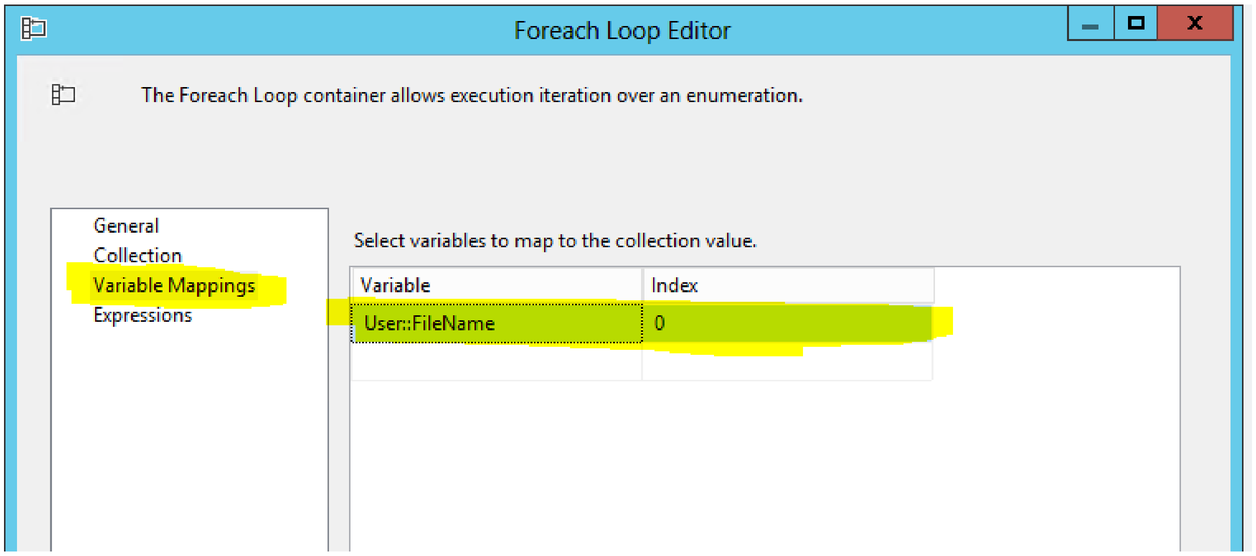
Foreach Loop Editor
Enumerator: Foreach File Enumerator
Expressions

Property Expressions Editor

Property expressions:

Property	Expression
Directory	@[User::SourceFolder]

b) Configure the variable mapping

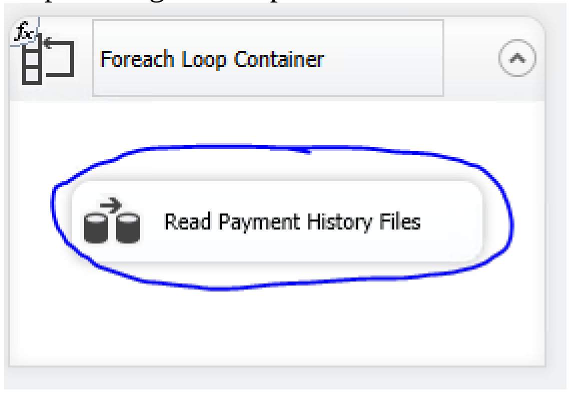


The Foreach Loop container allows execution iteration over an enumeration.

Select variables to map to the collection value.

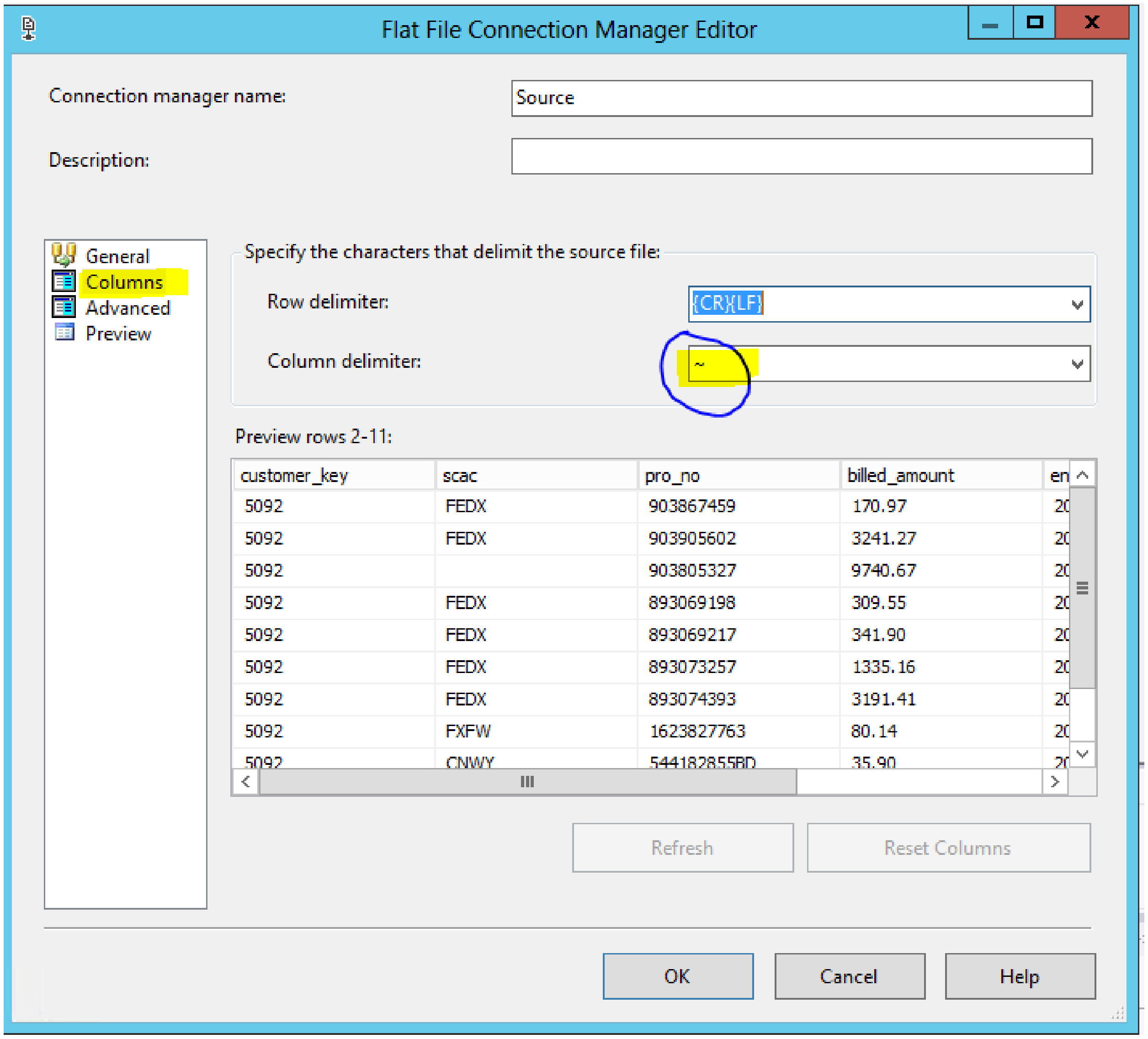
Variable	Index
User::FileName	0

Step 5: Drag and drop data flow task inside the for each loop container



Step 6: Create Source and Destination Connections

a) Create Flat File source for ~ Delimited Files



Connection manager name: Source

Description:

Specify the characters that delimit the source file:

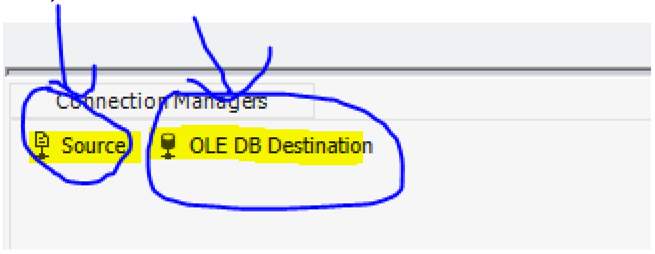
Row delimiter: {CR}{LF}

Column delimiter: ~

Preview rows 2-11:

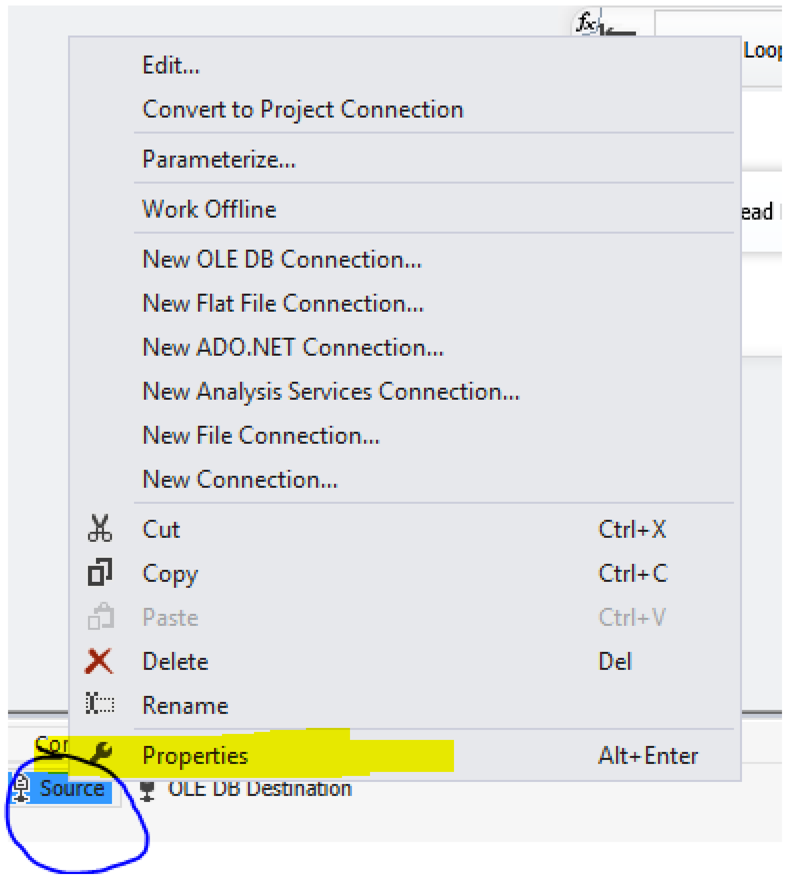
customer_key	scac	pro_no	billed_amount	en
5092	FEDX	903867459	170.97	20
5092	FEDX	903905602	3241.27	20
5092		903805327	9740.67	20
5092	FEDX	893069198	309.55	20
5092	FEDX	893069217	341.90	20
5092	FEDX	893073257	1335.16	20
5092	FEDX	893074393	3191.41	20
5092	FXFW	1623827763	80.14	20
5092	CNWY	544182855BD	35.90	20

b) Create OLD DB Destination

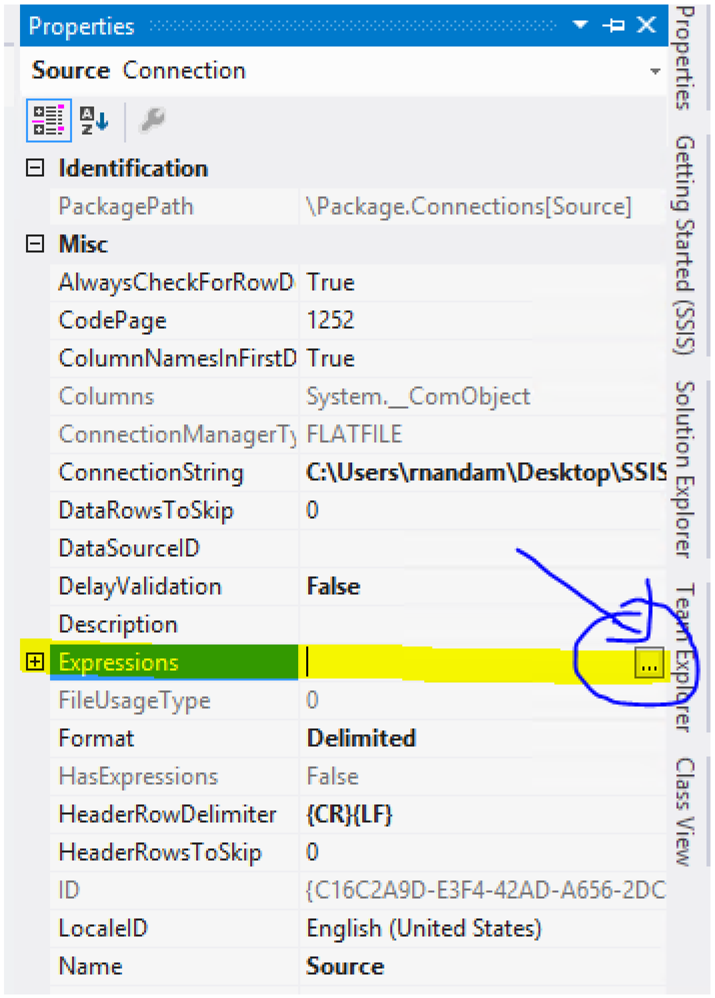


Step 7: Configure the Flat Source Dynamic Configuration for connection string expression

1) Right click on Flat File Source Connection then go to property window



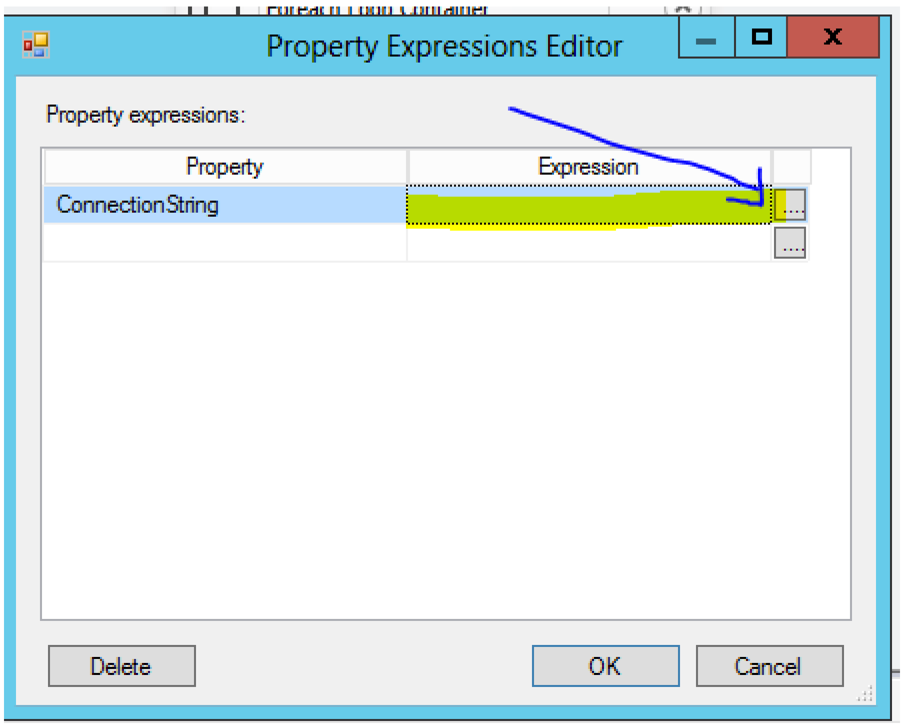
2) In the property window select expressions



Source Connection

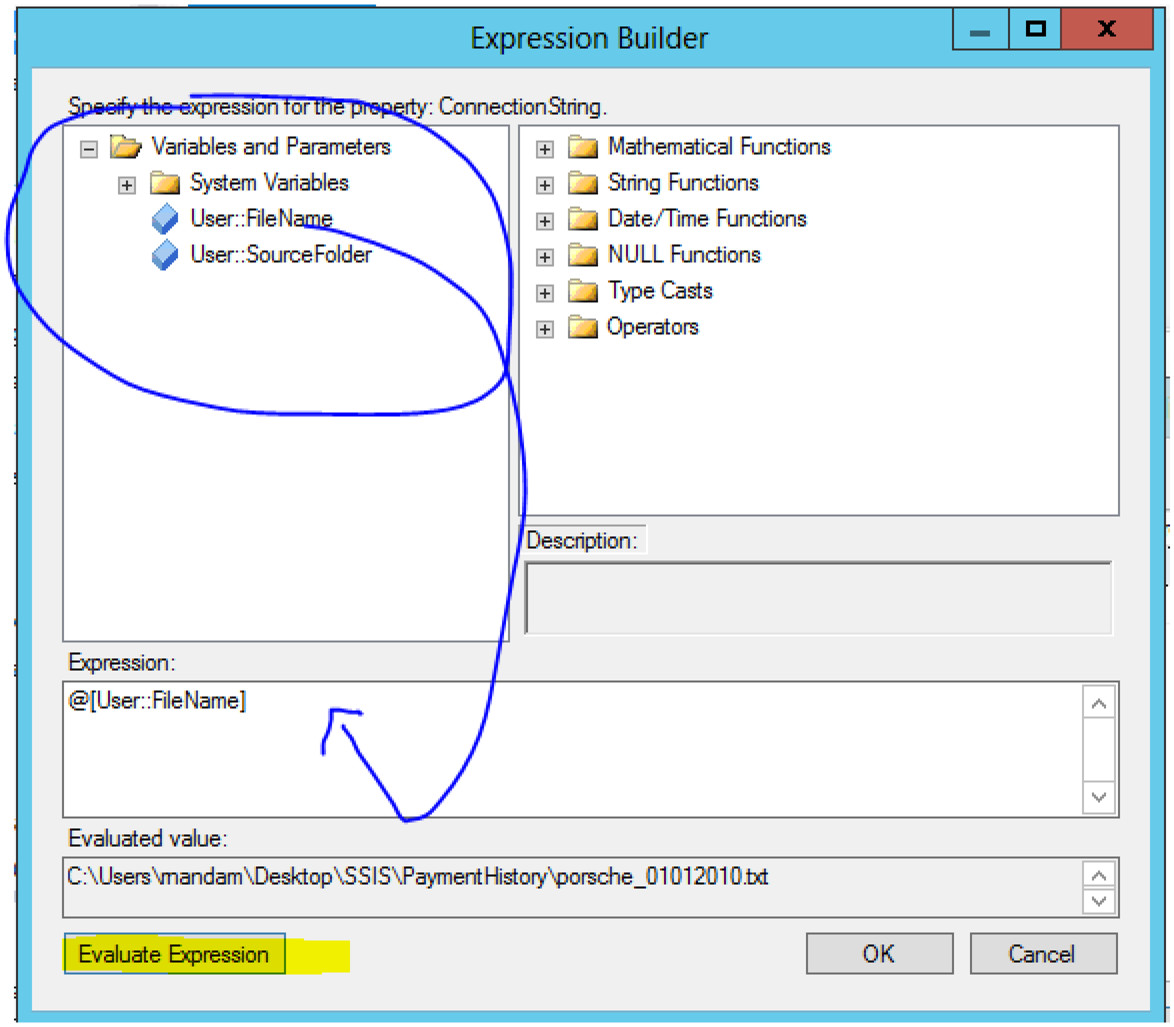
Identification	
PackagePath	\Package.Connections[Source]
Misc	
AlwaysCheckForRowD	True
CodePage	1252
ColumnNamesInFirstD	True
Columns	System.__ComObject
ConnectionManagerTy	FLATFILE
ConnectionString	**C:\Users\rnandam\Desktop\SSIS**
DataRowsToSkip	0
DataSourceID	
DelayValidation	**False**
Description	
Expressions	...
FileUsageType	0
Format	**Delimited**
HasExpressions	False
HeaderRowDelimiter	**{CR}{LF}**
HeaderRowsToSkip	0
ID	{C16C2A9D-E3F4-42AD-A656-2DC
LocaleID	English (United States)
Name	**Source**

3) In the expression editor select property as Connection string and assign File name variable for the expression

4) 

Property	Expression
ConnectionString	

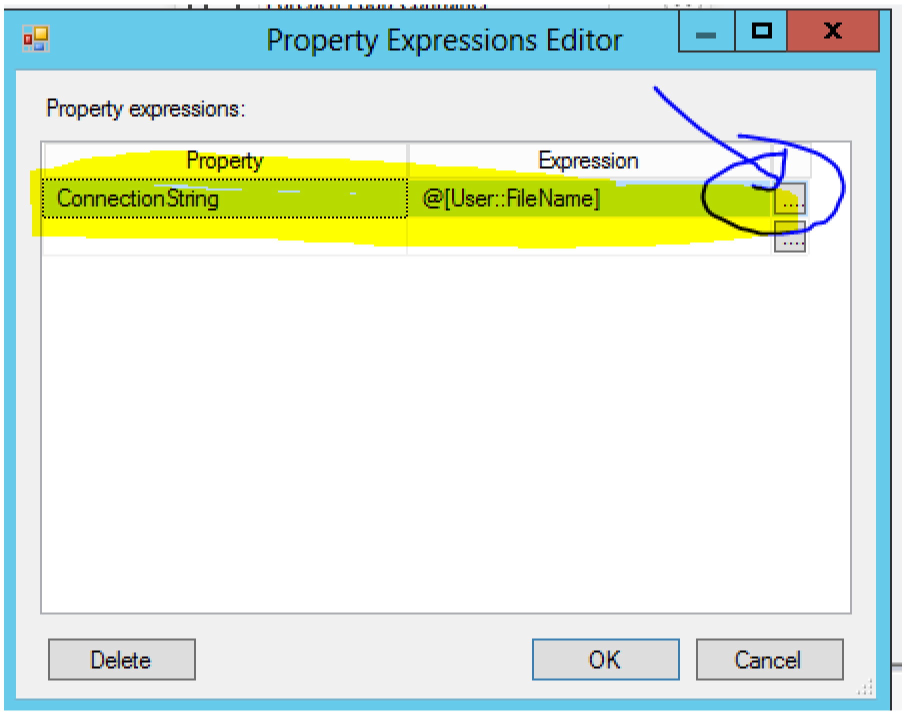
Then select File name variable from the Variable Collection



```
@[User::FileName]
```

```
C:\Users\mandam\Desktop\SSIS\PaymentHistory\porsche_01012010.txt
```

After it should look like this

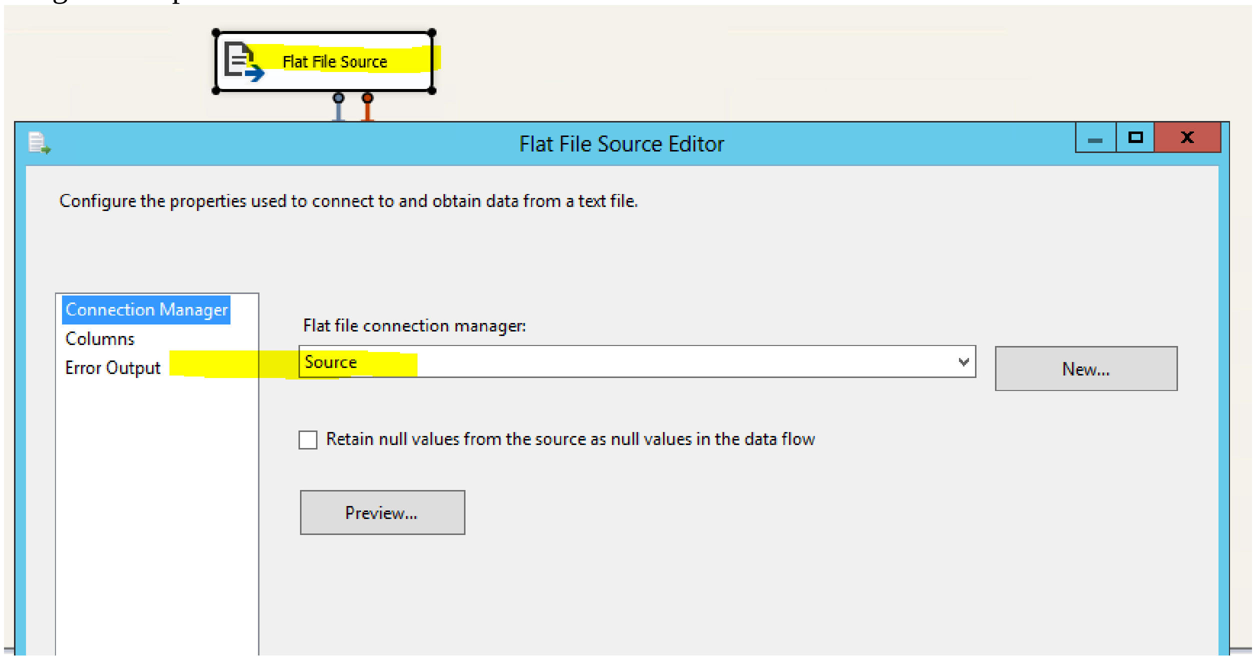


Property	Expression
ConnectionString	@[User::FileName]

Step 7: Configure the Data Flow task to read data from source to destination

Here we will do all business validations

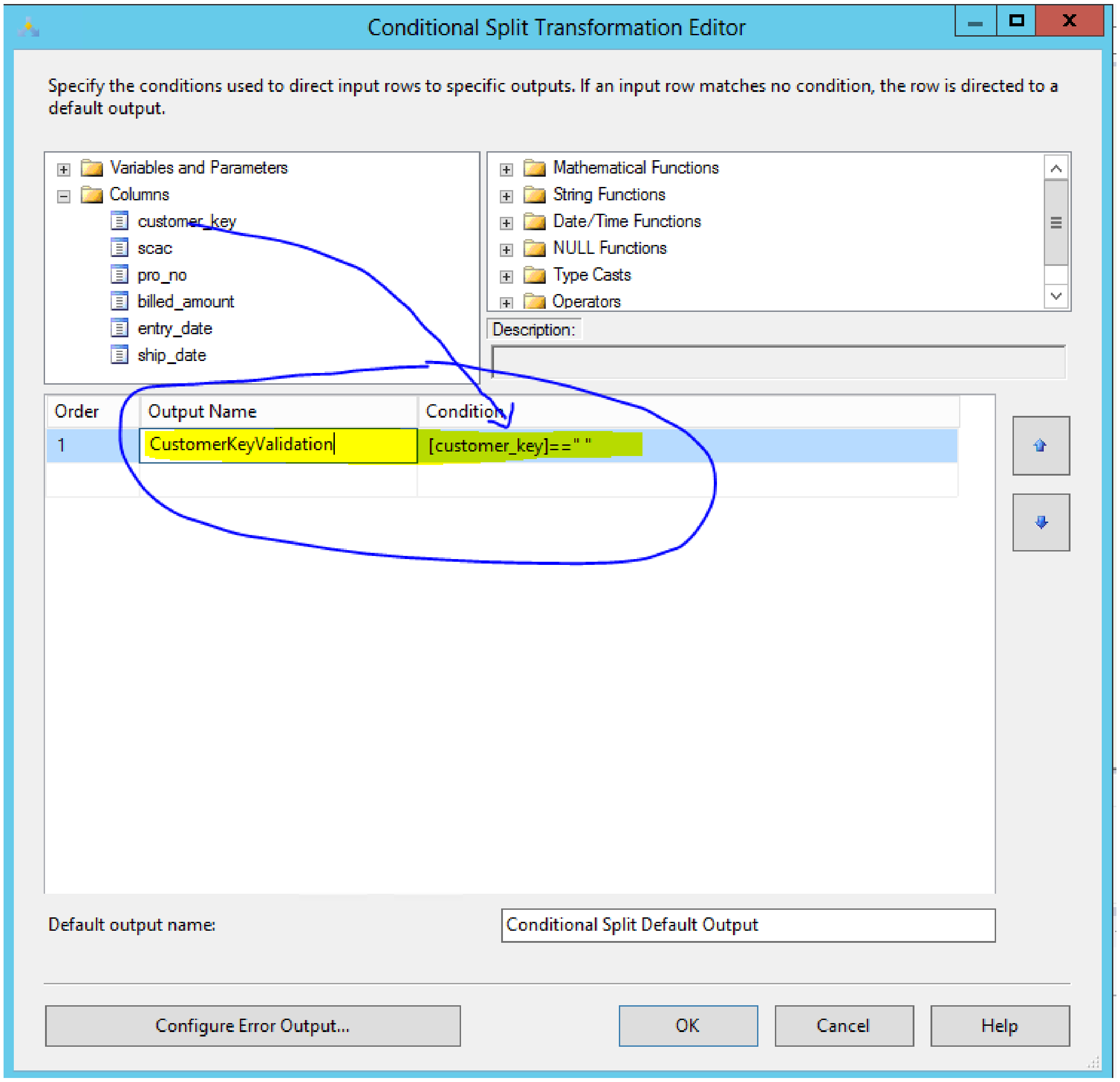
1) Drag and drop flat file source and select Flat File Connection


Flat File Source
Flat File Source Editor
Configure the properties used to connect to and obtain data from a text file.
Connection Manager
Columns
Error Output
New...
Retain null values from the source as null values in the data flow
Preview...

Flat file connection manager: Source

2) Use conditional split to do Customer key validation

Write expression in the conditional split to check does incoming record is missing any Customer Key values (I am using customer _key == “ ”) Expression

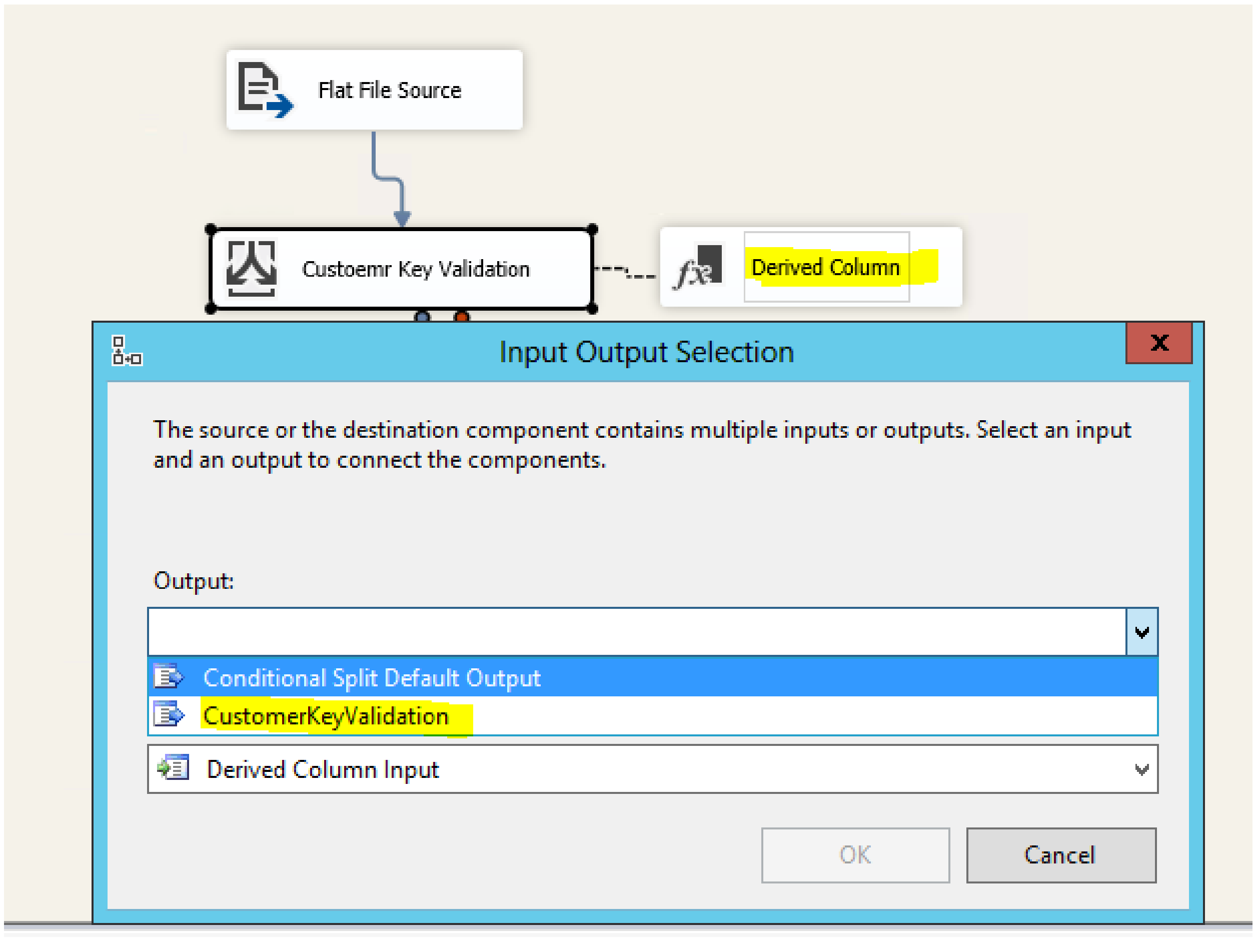

Conditional Split Transformation Editor

Specify the conditions used to direct input rows to specific outputs. If an input row matches no condition, the row is directed to a default output.

Order	Output Name	Condition
1	CustomerKeyValidation	[customer_key]==" "

Default output name: Conditional Split Default Output

3) Use Derived column transformation to add error message for matched output for Customer key validation output



The source or the destination component contains multiple inputs or outputs. Select an input and an output to connect the components.

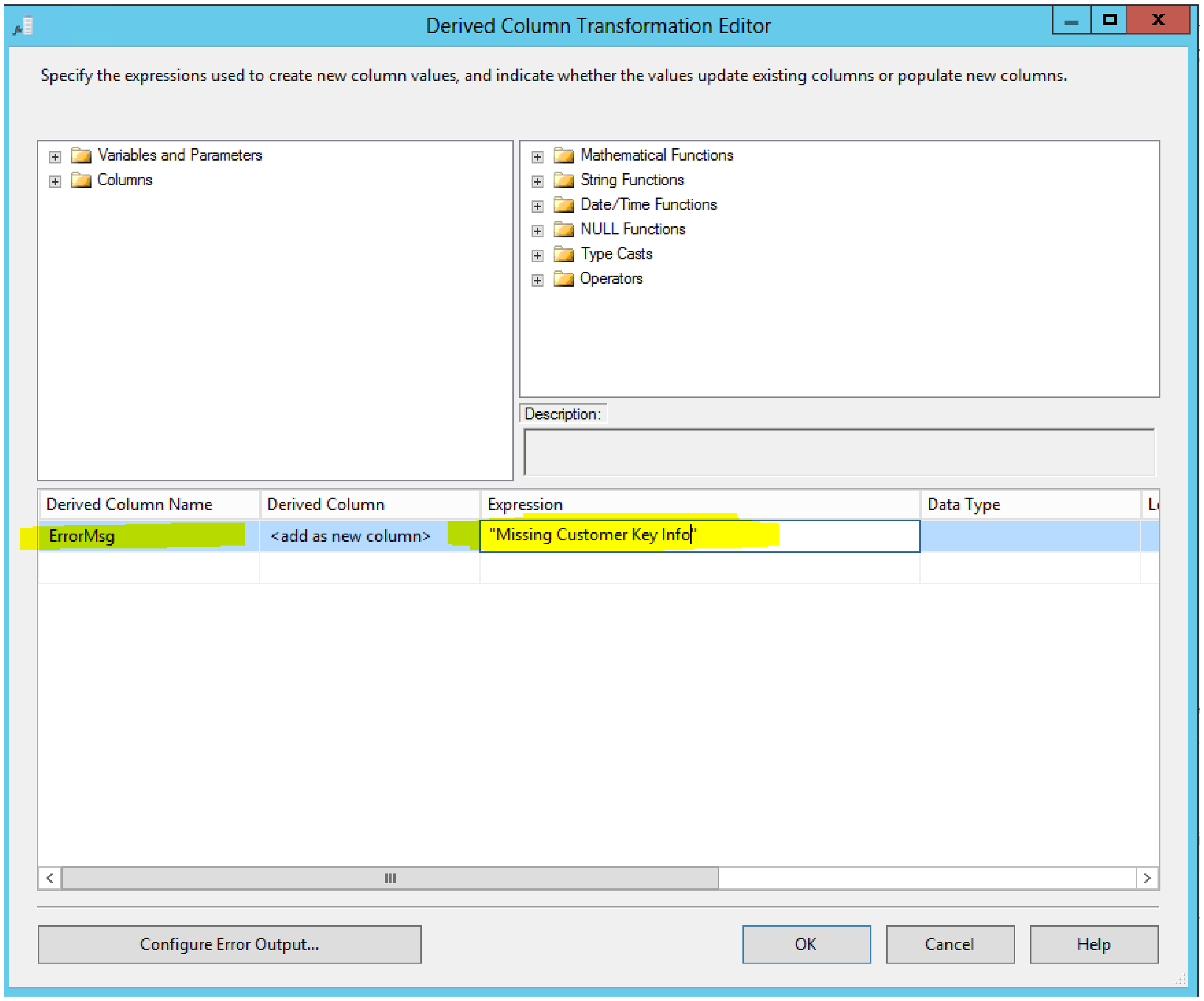
Output:

Conditional Split Default Output

CustomerKeyValidation

Derived Column Input

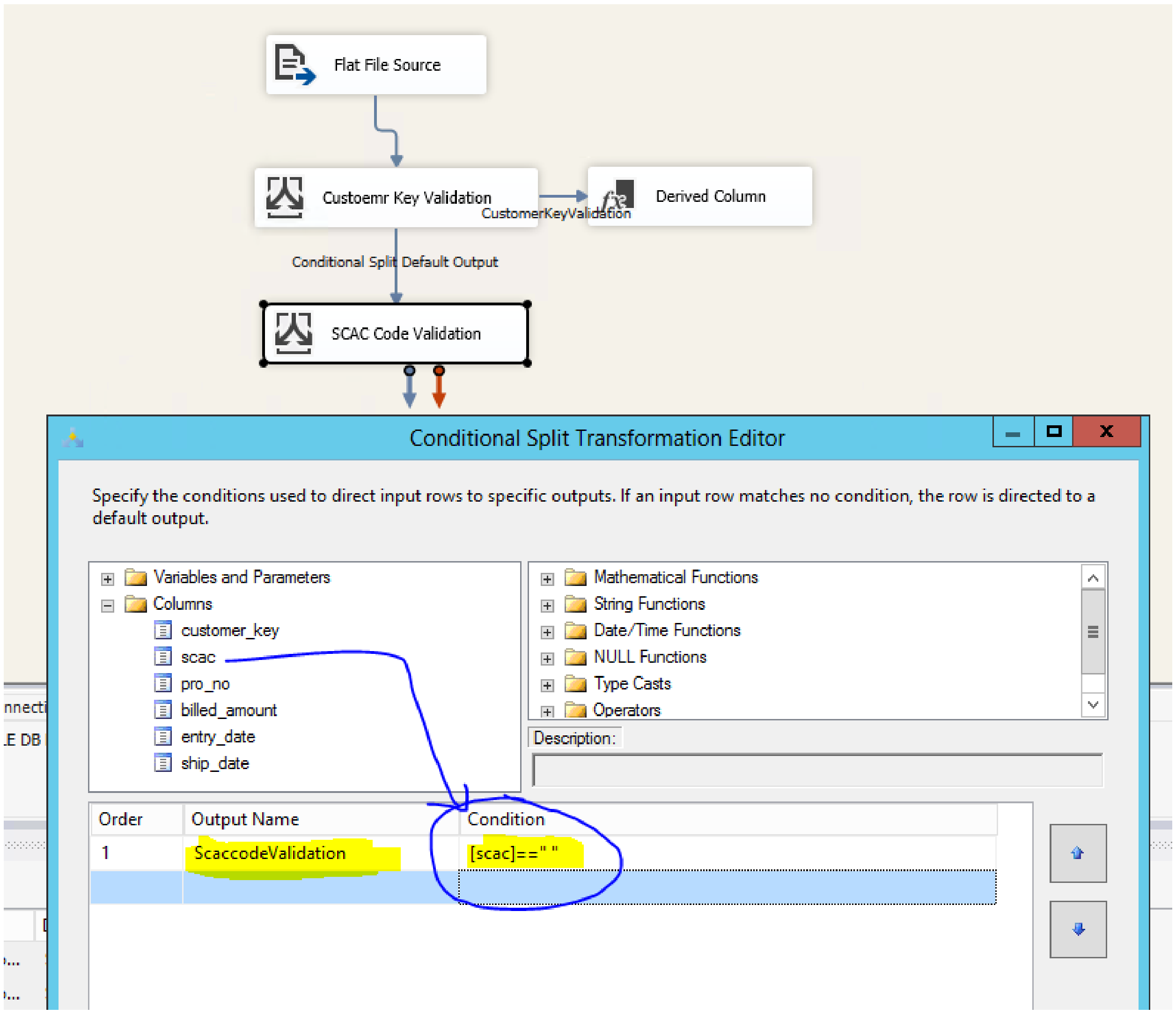
4) Add error message in the Derived column expression



Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Derived Column Name	Derived Column	Expression	Data Type
ErrorMsg	<add as new column>	"Missing Customer Key Info"	

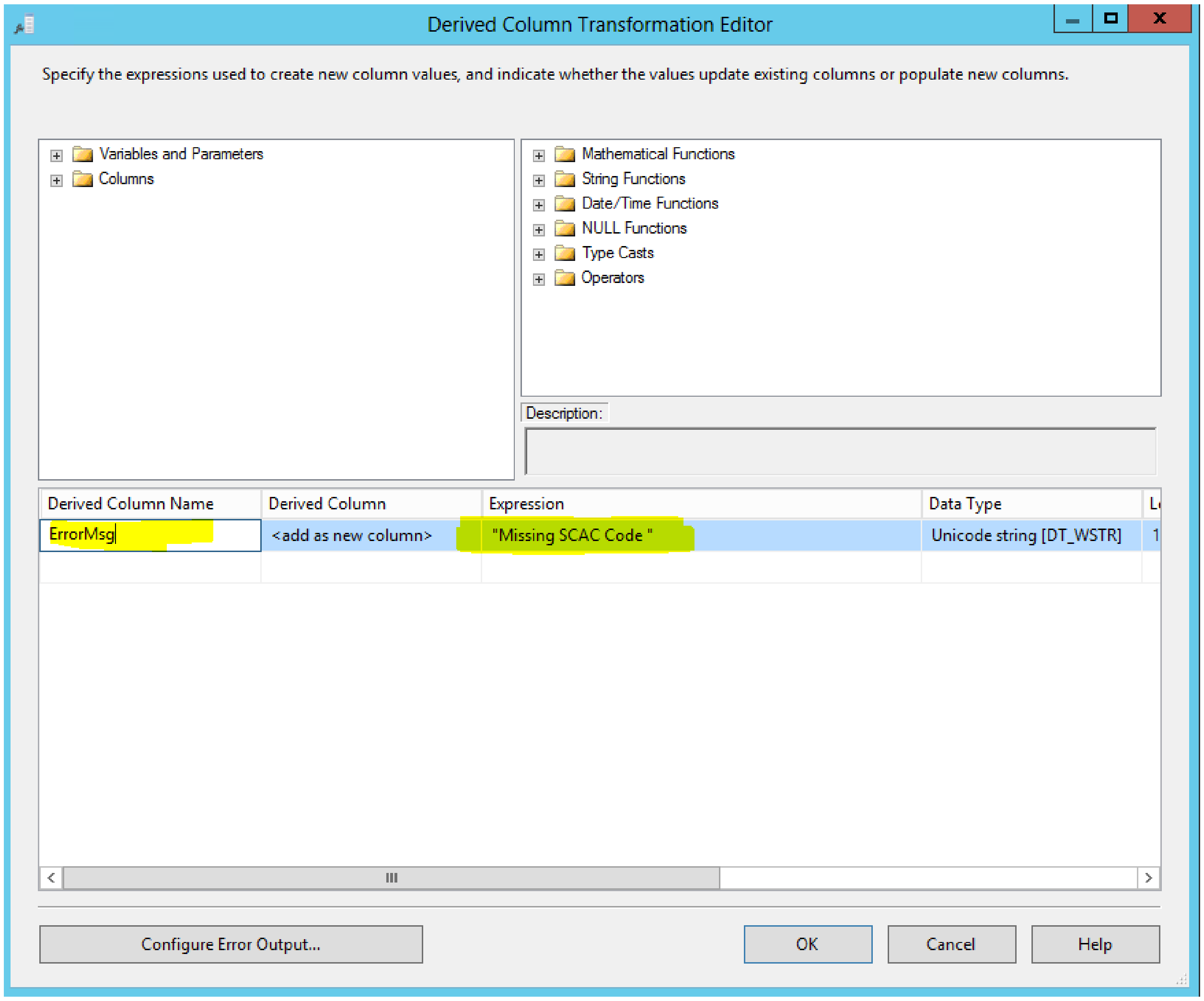
5) Implement conditional split for the SCAC Code validation



Specify the conditions used to direct input rows to specific outputs. If an input row matches no condition, the row is directed to a default output.

Order	Output Name	Condition
1	ScaccodeValidation	[scac]==" "

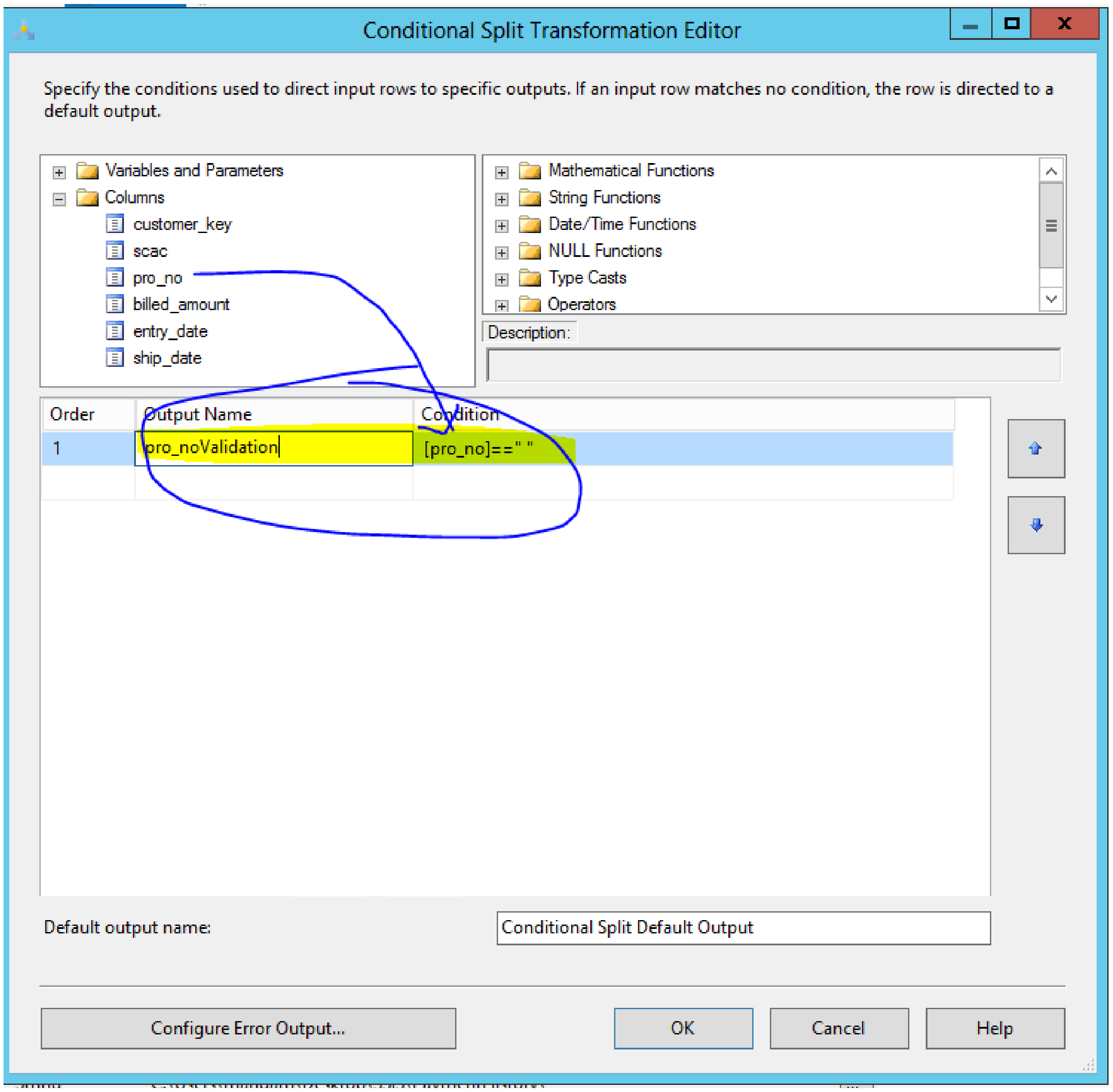
6) Add Derived column for missing scac code output



Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Derived Column Name	Derived Column	Expression	Data Type	L
ErrorMsg	<add as new column>	"Missing SCAC Code "	Unicode string [DT_WSTR]	1

7) Pro no validation



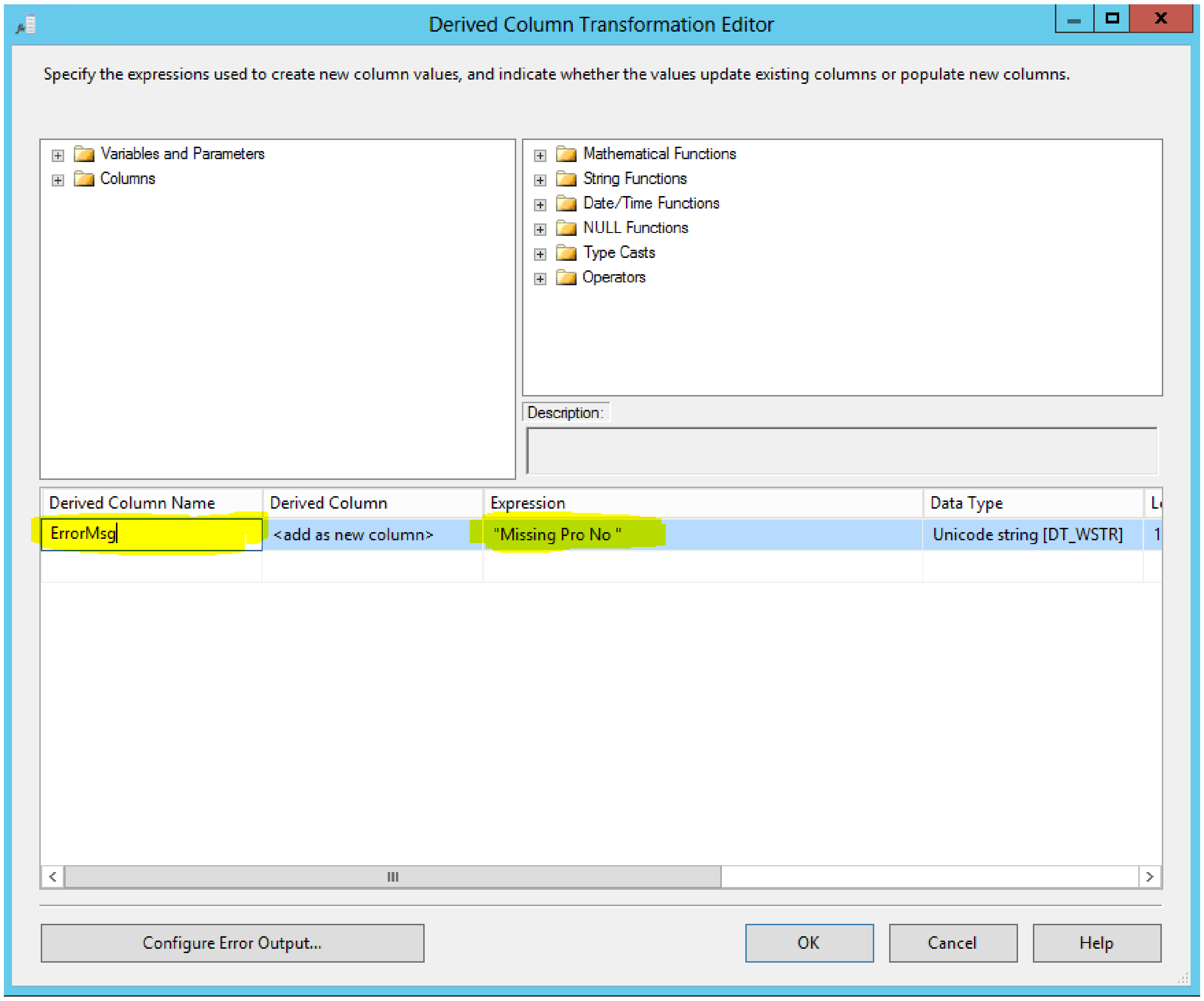
Conditional Split Transformation Editor

Specify the conditions used to direct input rows to specific outputs. If an input row matches no condition, the row is directed to a default output.

Order	Output Name	Condition
1	pro_noValidation	[pro_no]==" "

Default output name: Conditional Split Default Output

8) Adding error msg for pro no in the derived column

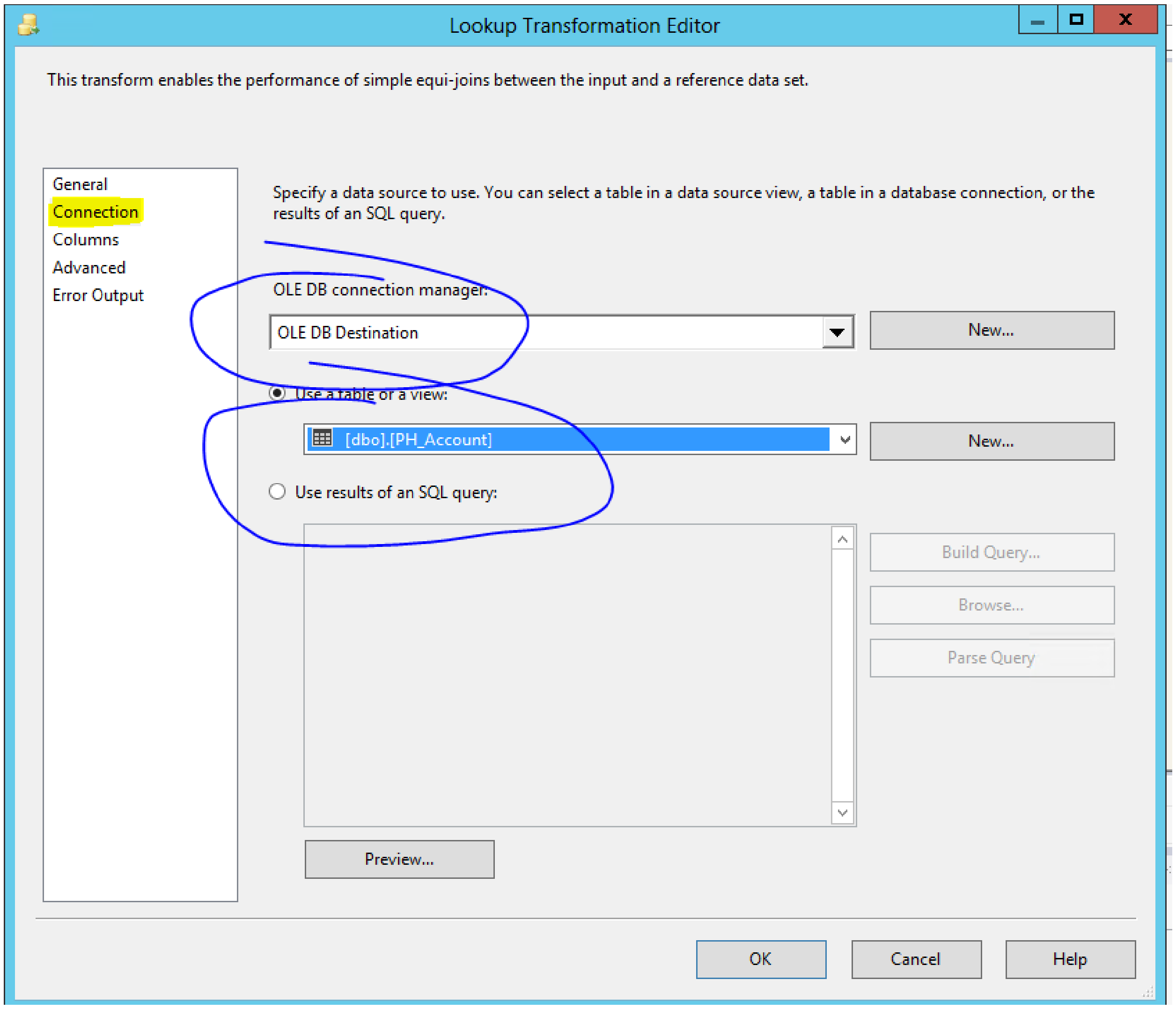


Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Derived Column Name	Derived Column	Expression	Data Type	L
ErrorMsg	<add as new column>	"Missing Pro No "	Unicode string [DT_WSTR]	1

Step 8: Check Does record exist in the Destination or Not (If exist we need to update else needs to insert the record)

Drag and drop look up transformation



Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

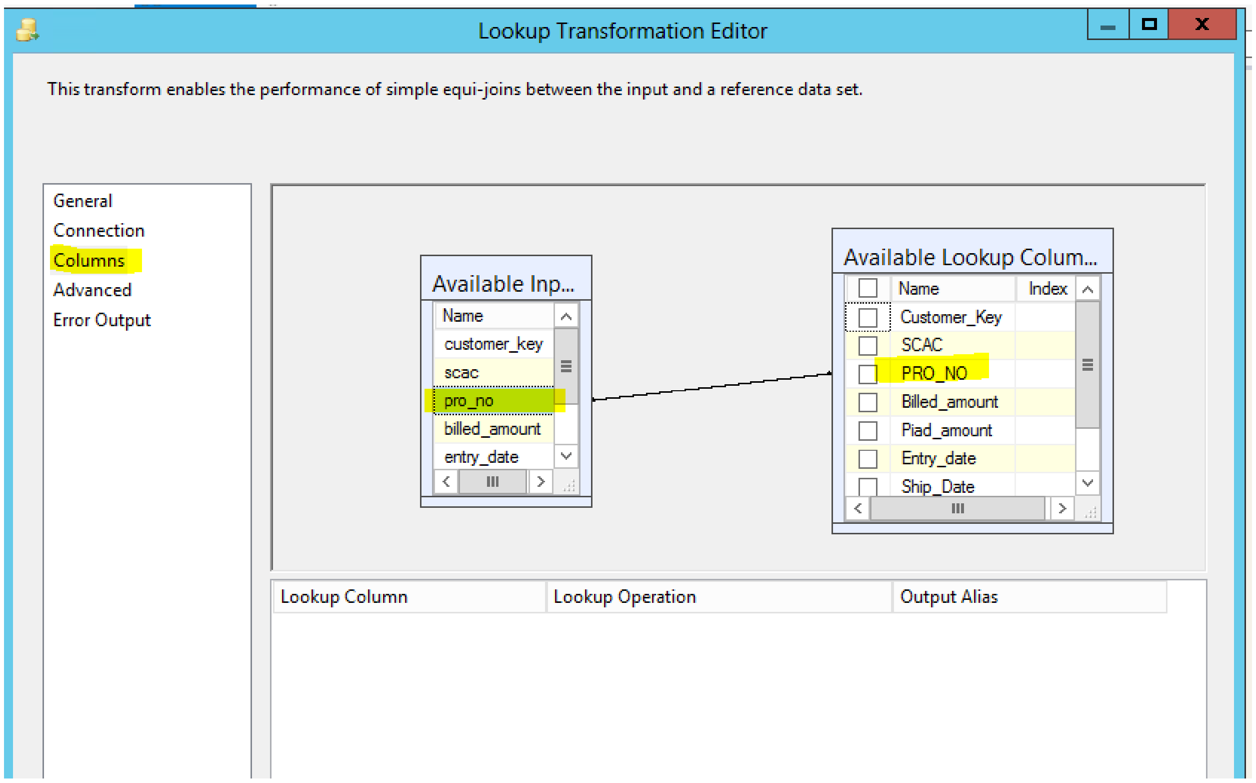
Specify a data source to use. You can select a table in a data source view, a table in a database connection, or the results of an SQL query.

OLE DB connection manager: OLE DB Destination

[✓ Use a table or a view:] [dbo].[PH_Account]

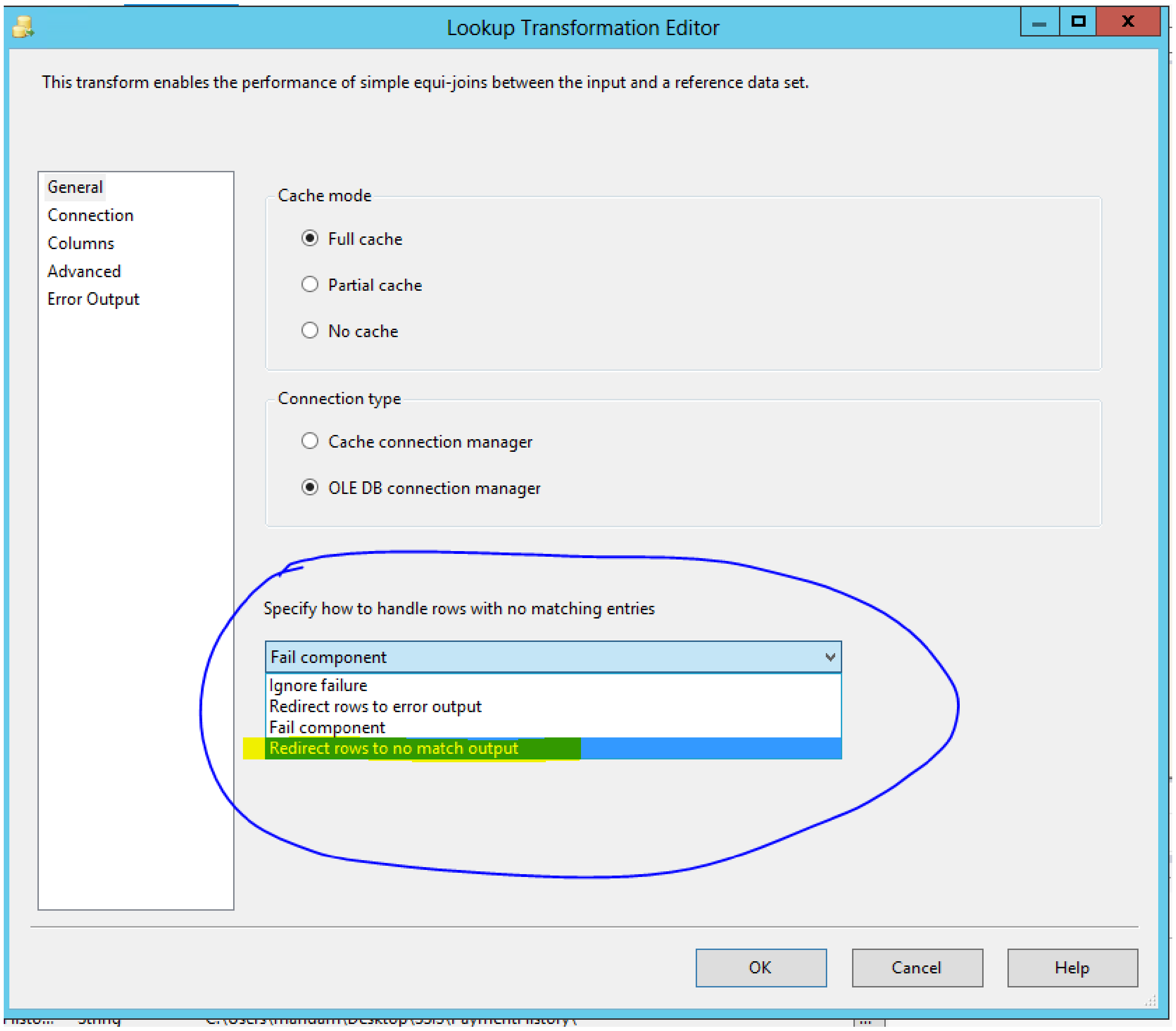
Use results of an SQL query:

Then click on the Columns


Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

Then select Specify how to handle rows with no matching entries



This transform enables the performance of simple equi-joins between the input and a reference data set.

Cache mode

- (●) Full cache
- ( ) Partial cache
- ( ) No cache

Connection type

- ( ) Cache connection manager
- (●) OLE DB connection manager

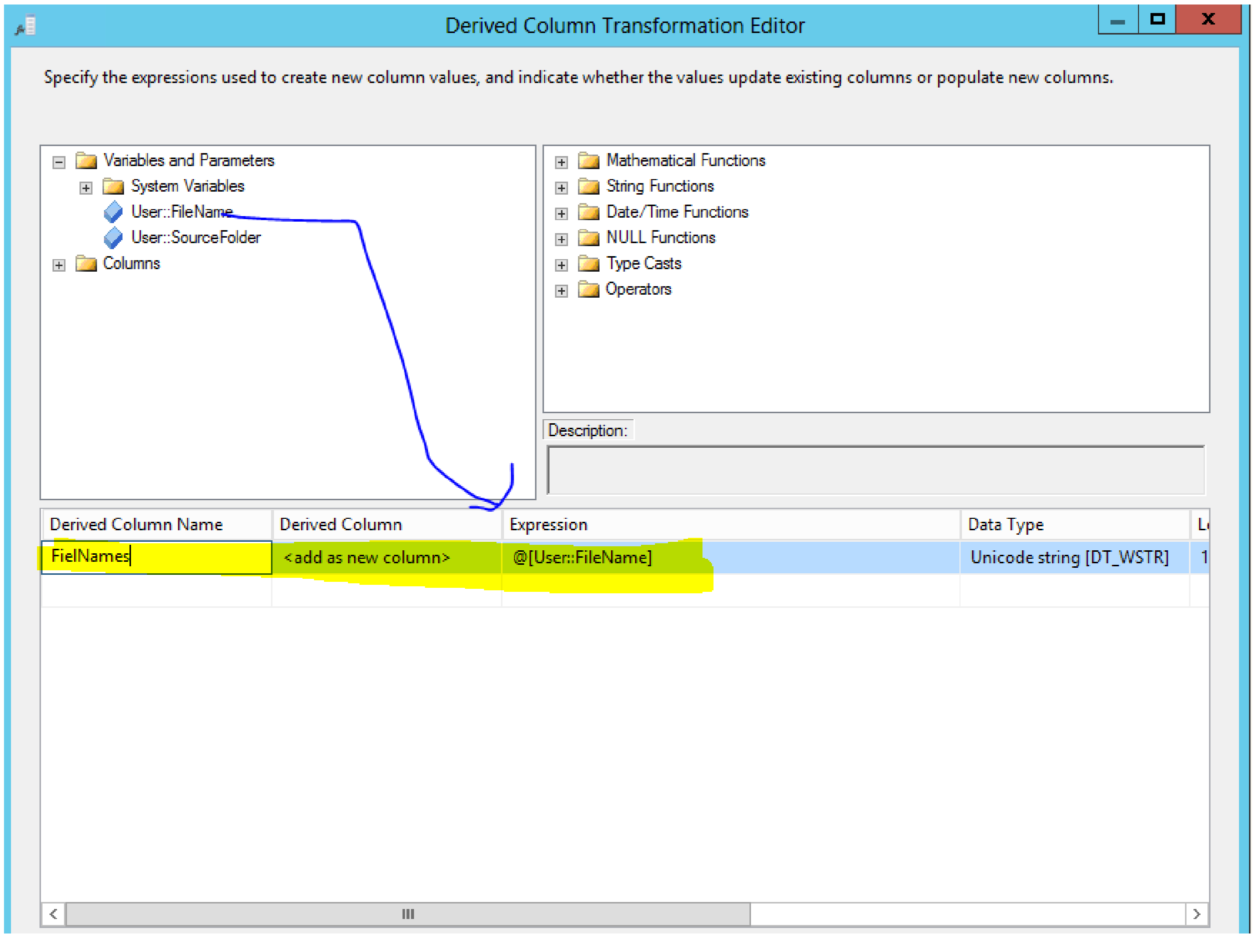
Specify how to handle rows with no matching entries

Fail component

- Ignore failure
- Redirect rows to error output
- Fail component
- Redirect rows to no match output

Now take no match output records and load to Destination table as record not exist in the destination table

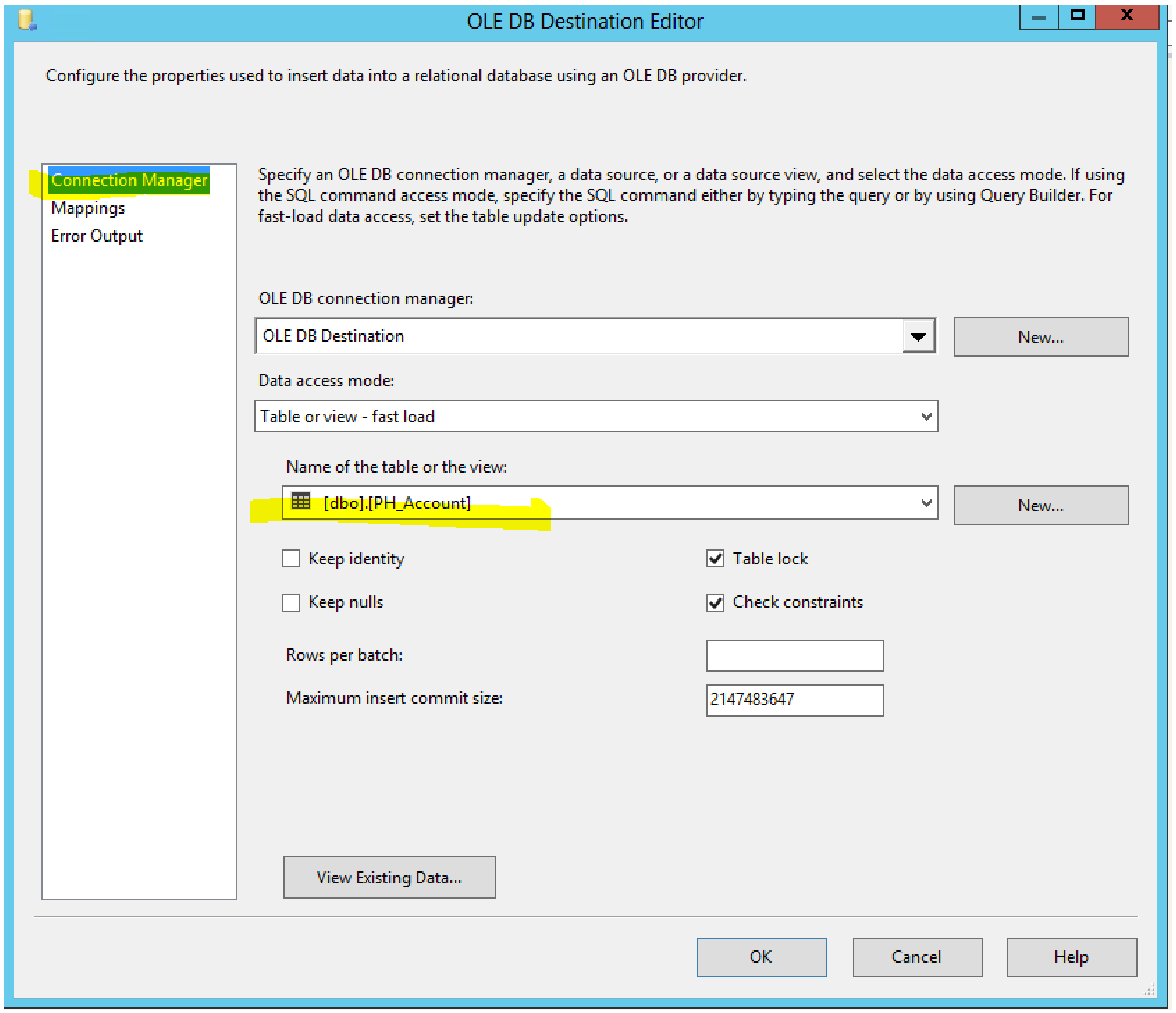
Before that we need to add File name to destination stream
Drag and drop Derived column to add file name



Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

Derived Column Name	Derived Column	Expression	Data Type	L
FielNames	<add as new column>	@[User::FileName]	Unicode string [DT_WSTR]	1

Then map not matched records to destination table



Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: OLE DB Destination

Data access mode: Table or view - fast load

Name of the table or the view: [dbo].[PH_Account]

☐ Keep identity
☑ Table lock
☐ Keep nulls
☑ Check constraints

Rows per batch:

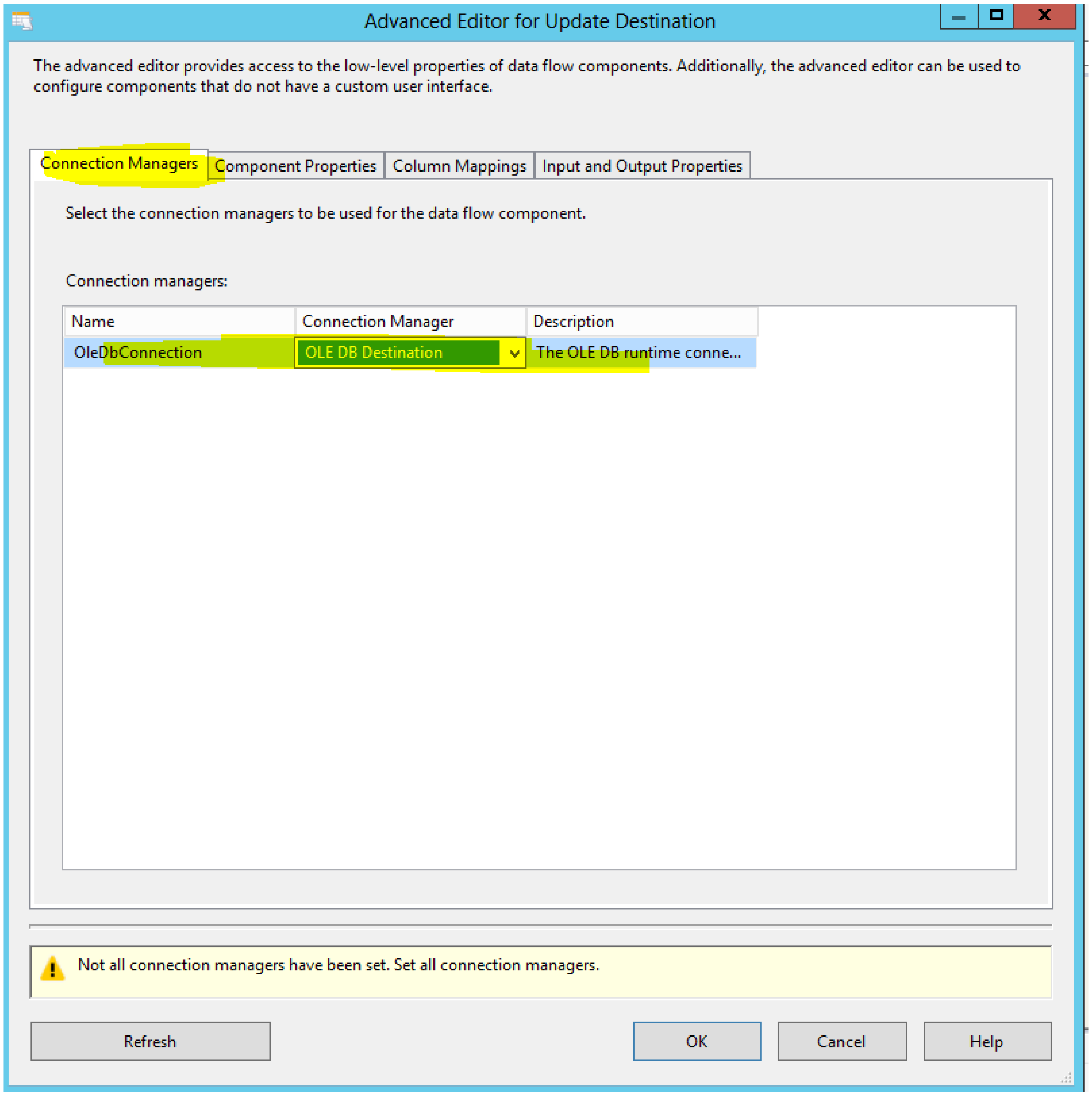
Maximum insert commit size: 2147483647

Now if record exists in the destination table we need to update that record with correct billed amount

Use OLE DB Command to update the destination table

Drag and drop OLE DB Command for matched output from look up

a) Select Connection Manager in the OLEDB Command



Advanced Editor for Update Destination

The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Select the connection managers to be used for the data flow component.

Connection managers:

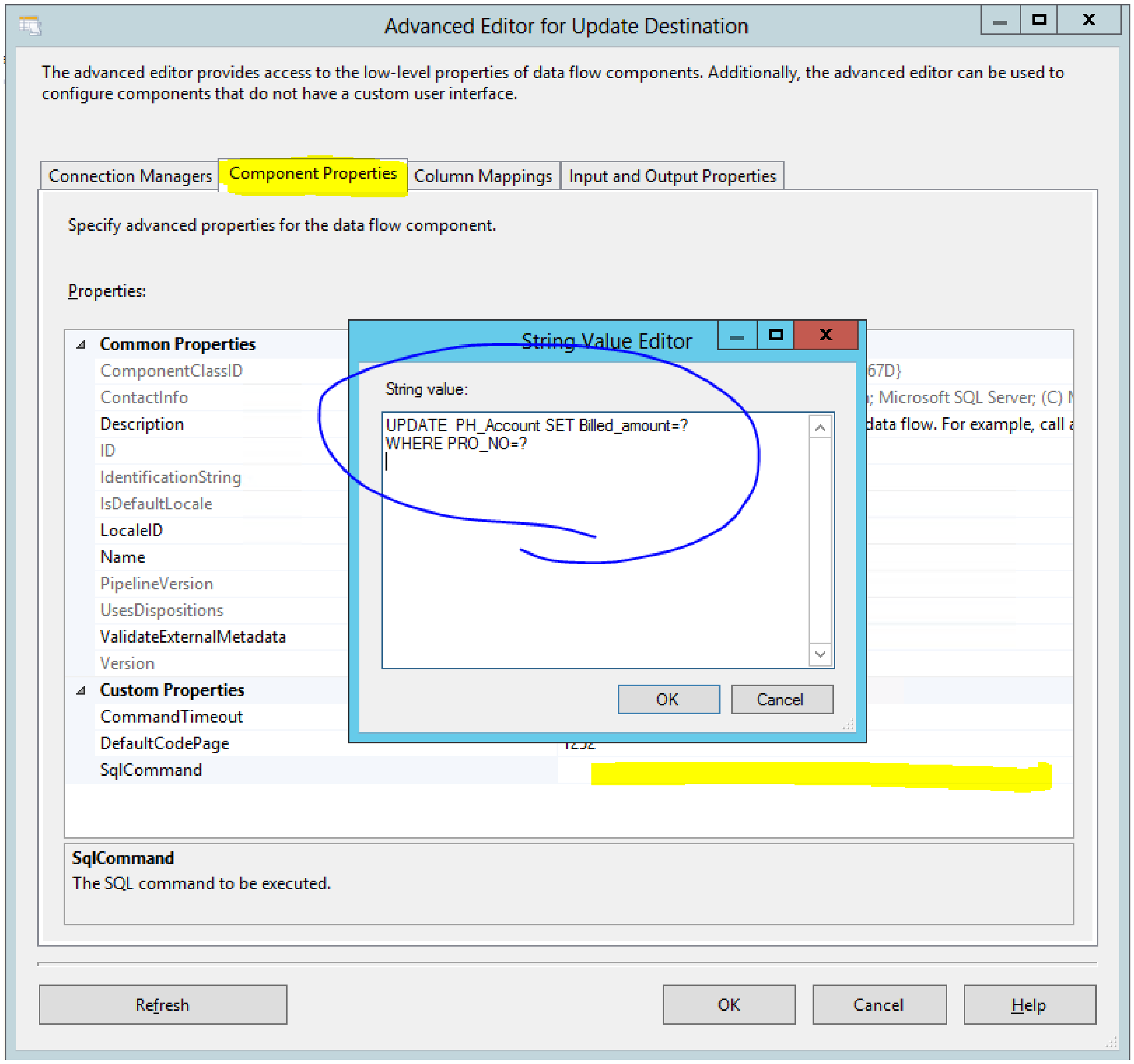
Name	Connection Manager	Description
OleDbConnection	OLE DB Destination	The OLE DB runtime conne...

Not all connection managers have been set. Set all connection managers.

b) Provide the Update Statement in the Component property

Update statement

```
UPDATE  PH_Account SET Billed_amount=?
WHERE PRO_NO=?
```


Advanced Editor for Update Destination

The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Connection Managers | Component Properties | Column Mappings | Input and Output Properties

Specify advanced properties for the data flow component.

Properties:

Common Properties
ComponentClassID
ContactInfo
Description
ID
IdentificationString
IsDefaultLocale
LocaleID
Name
PipelineVersion
UsesDispositions
ValidateExternalMetadata
Version
Custom Properties
CommandTimeout
DefaultCodePage
SqlCommand

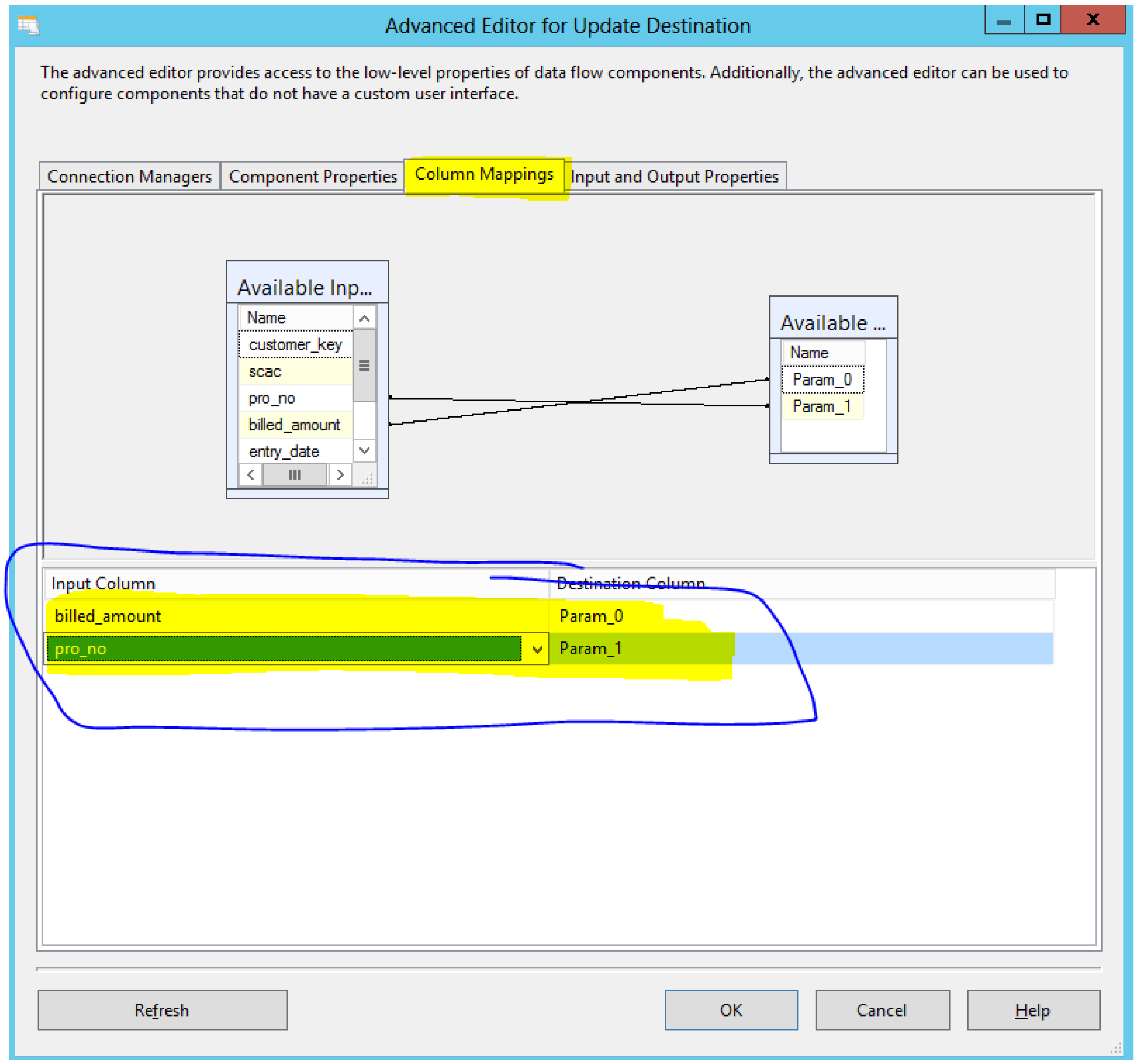
String Value Editor

String value:

```
UPDATE  PH_Account SET Billed_amount=?
WHERE PRO_NO=?
```

SqlCommand
The SQL command to be executed.

c) Now Map the Columns between source and Destination T SQL Statement

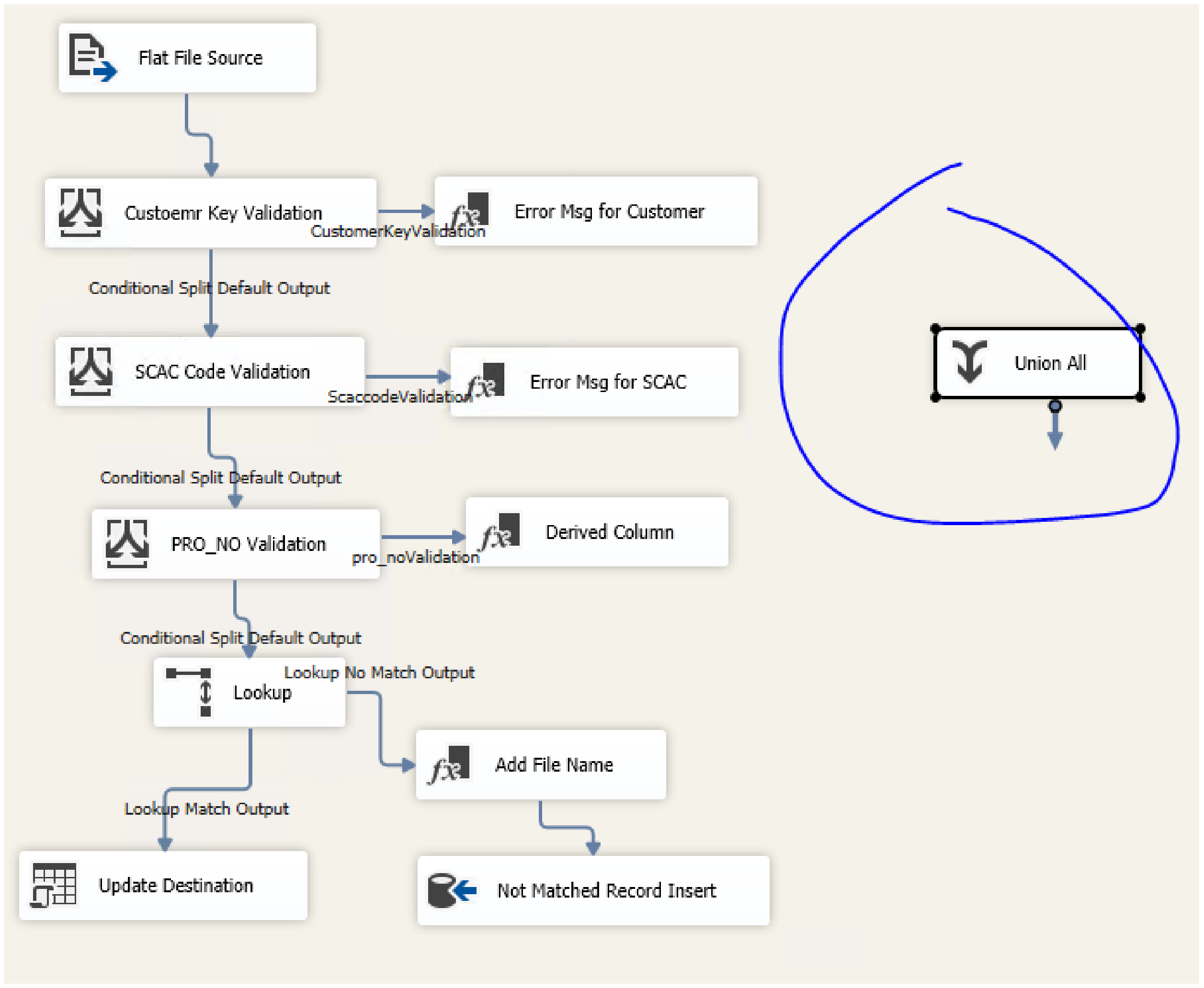


The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

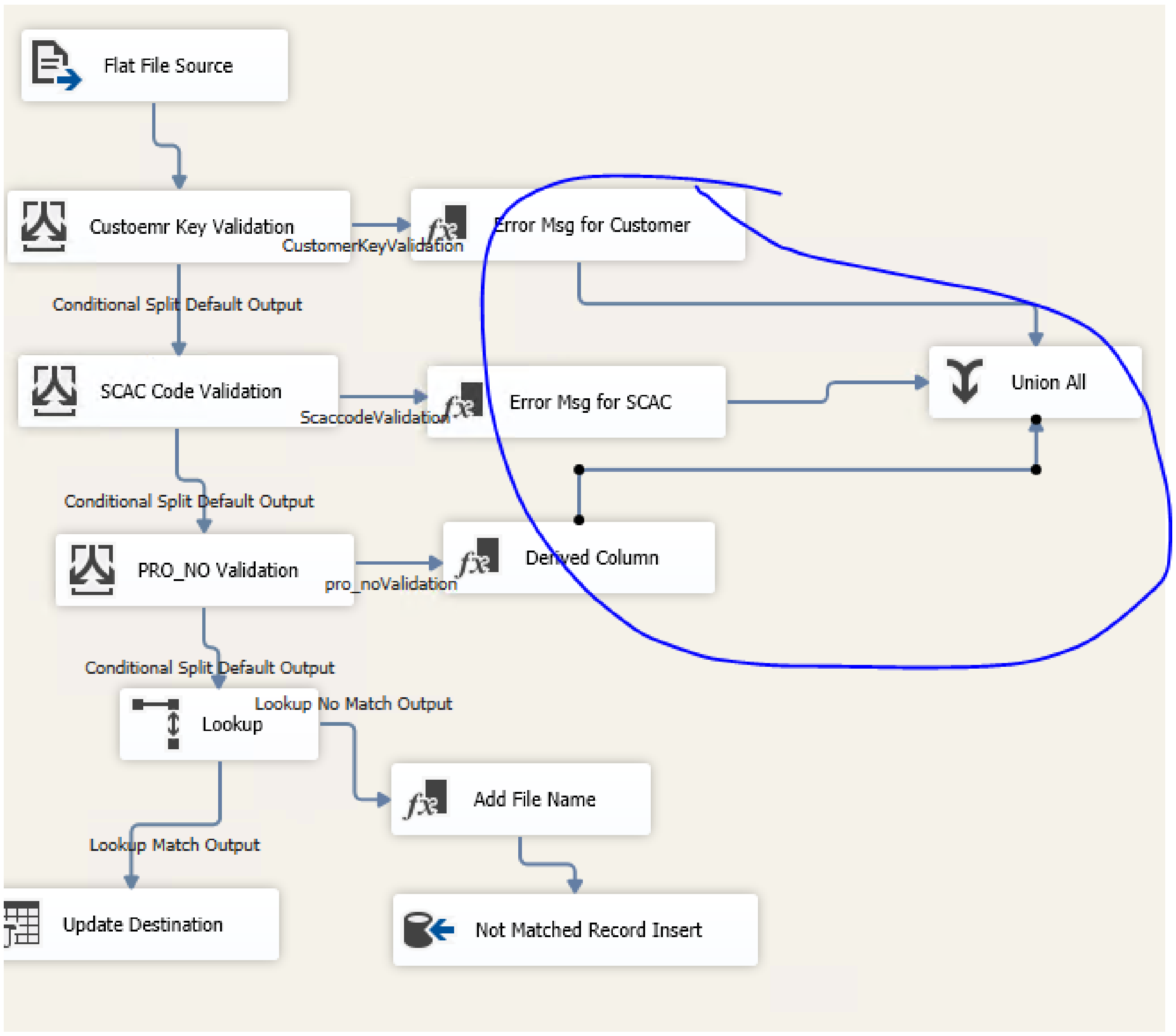
Input Column	Destination Column
billed_amount	Param_0
pro_no	Param_1

Step 9: Now combine all the error records into single data set using the Union all Transformation

a) Drag and drop Union all transformation

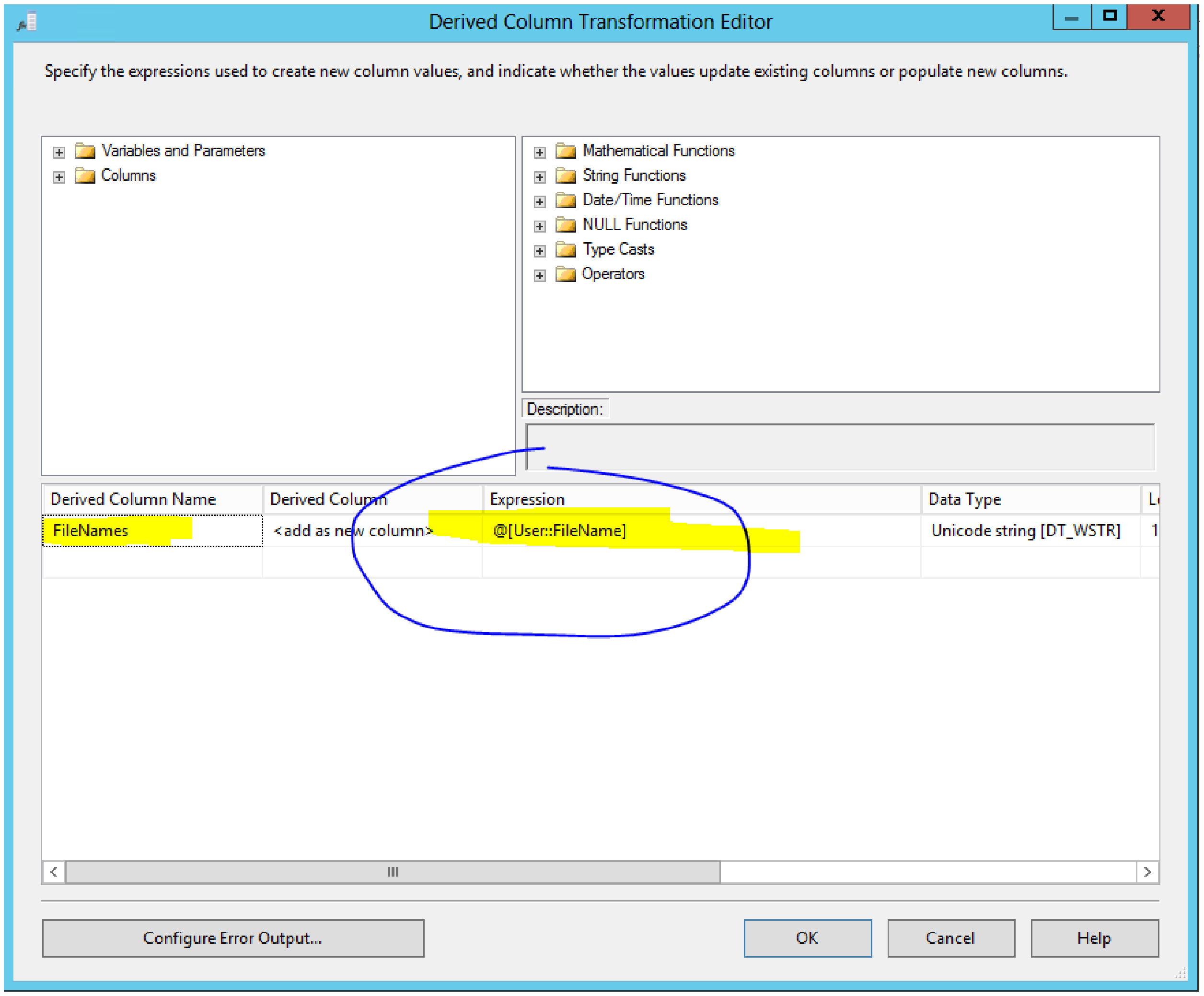

Flat File Source
Custoemr Key Validation
CustomerKeyValidation
Error Msg for Customer
Conditional Split Default Output
SCAC Code Validation
ScaccodeValidation
Error Msg for SCAC
Conditional Split Default Output
PRO_NO Validation
pro_noValidation
Derived Column
Conditional Split Default Output
Lookup
Lookup No Match Output
Add File Name
Lookup Match Output
Update Destination
Not Matched Record Insert
Union All

Then map it



Step 10) Add File Name to the error Output

a) Drag and drop derived column and add file name variable

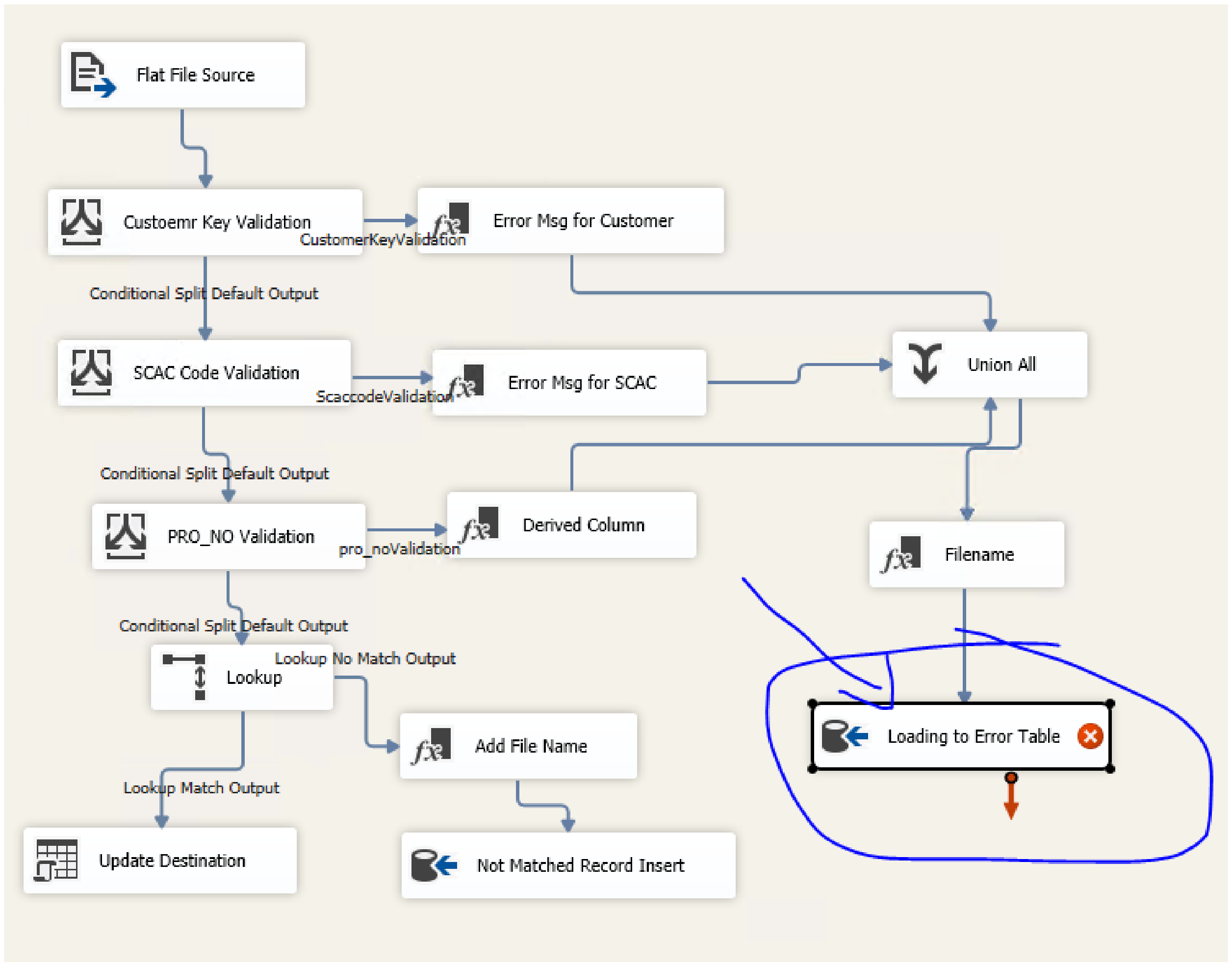


Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

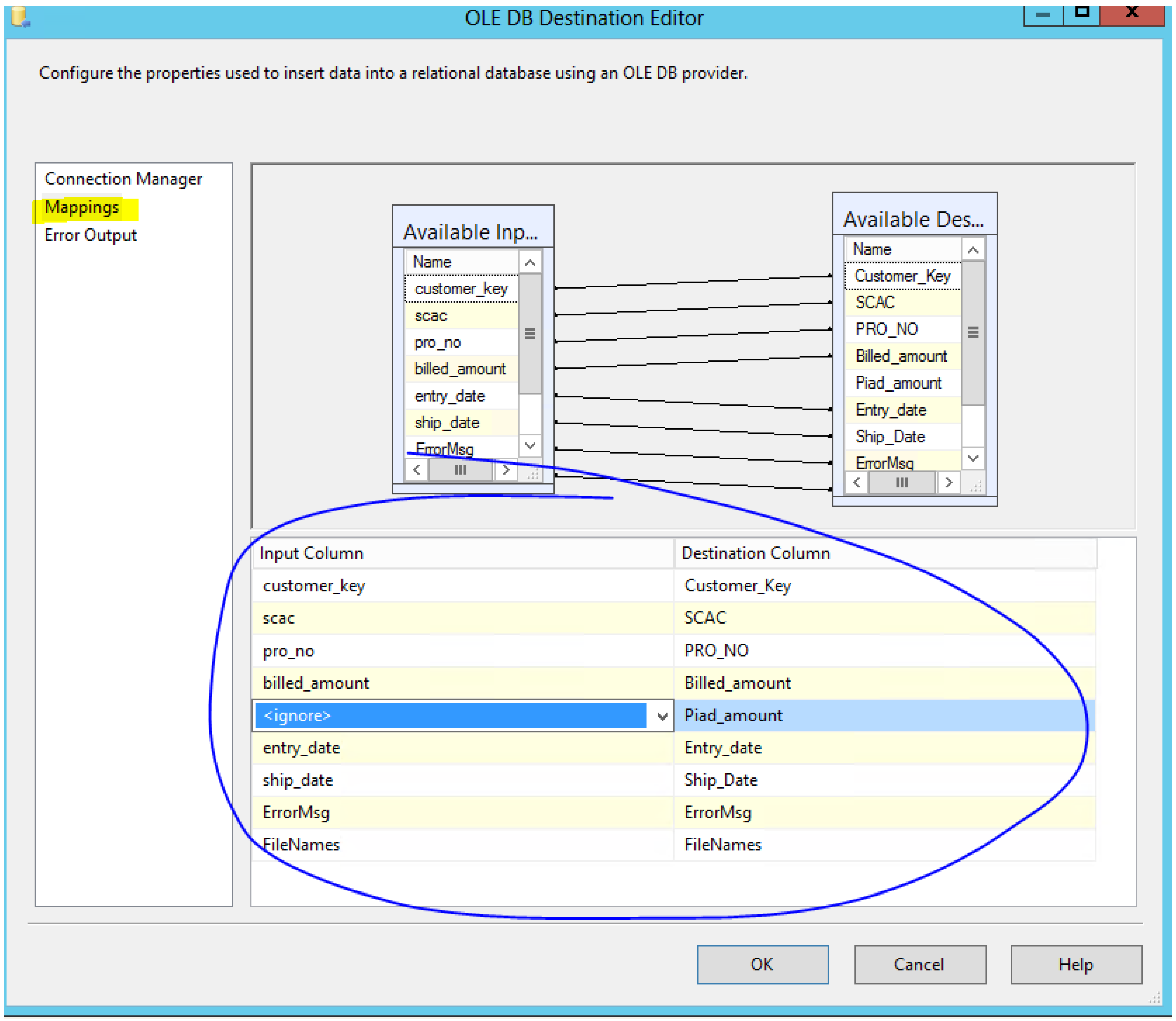
Derived Column Name	Derived Column	Expression	Data Type	L
FileNames	<add as new column>	@[User::FileName]	Unicode string [DT_WSTR]	1

Step 11) load All the error records to Error Table

A) **Drag and drop OLE DB Destination**



Now map all error records to error table



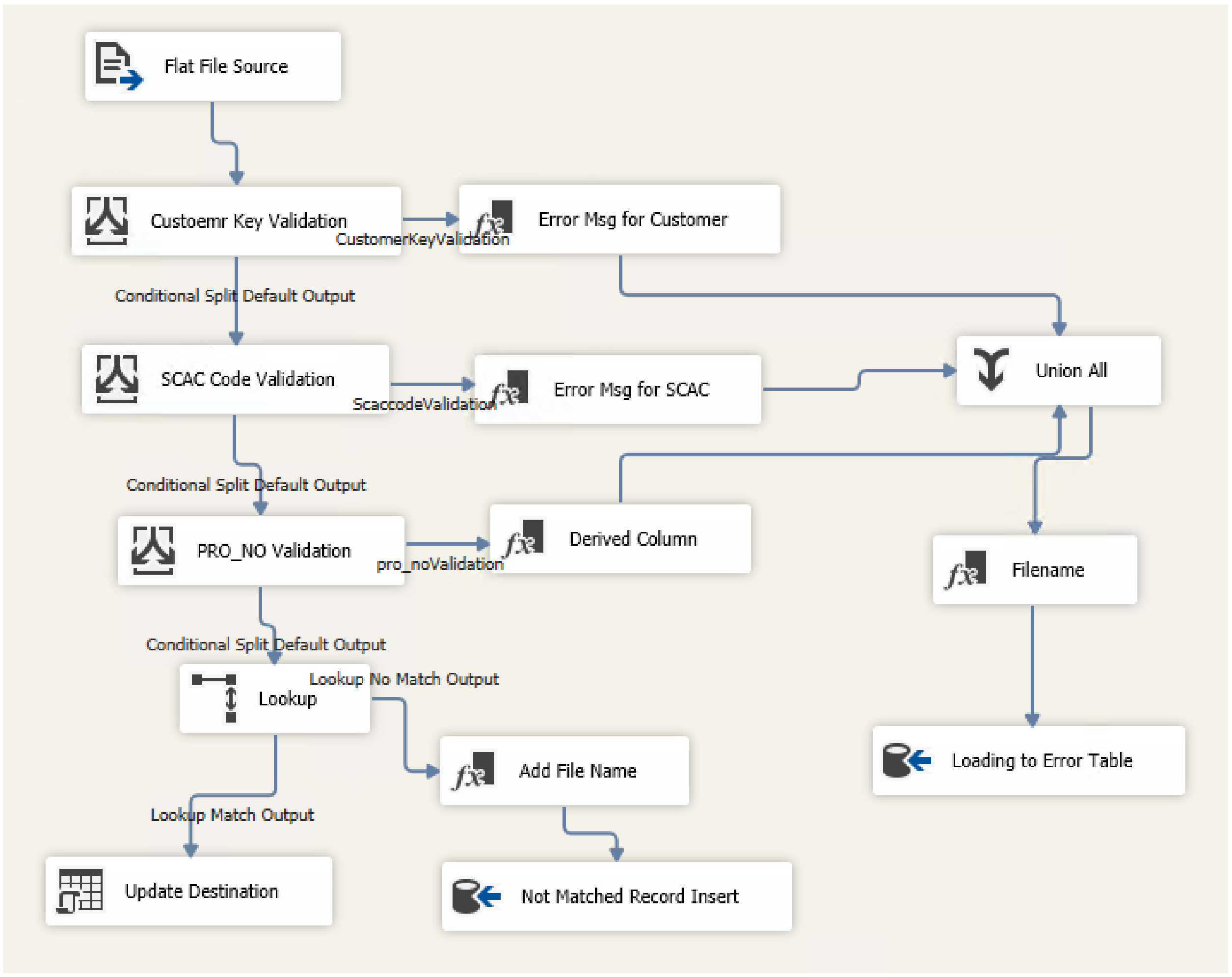
Configure the properties used to insert data into a relational database using an OLE DB provider.

Input Column	Destination Column
customer_key	Customer_Key
scac	SCAC
pro_no	PRO_NO
billed_amount	Billed_amount
<ignore>	Piad_amount
entry_date	Entry_date
ship_date	Ship_Date
ErrorMsg	ErrorMsg
FileNames	FileNames

Finally…

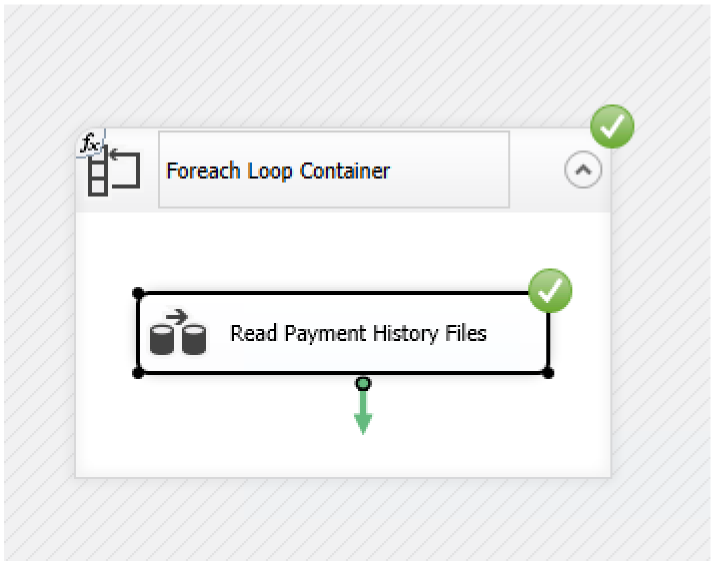
Now package should look like this

Data Flow Task



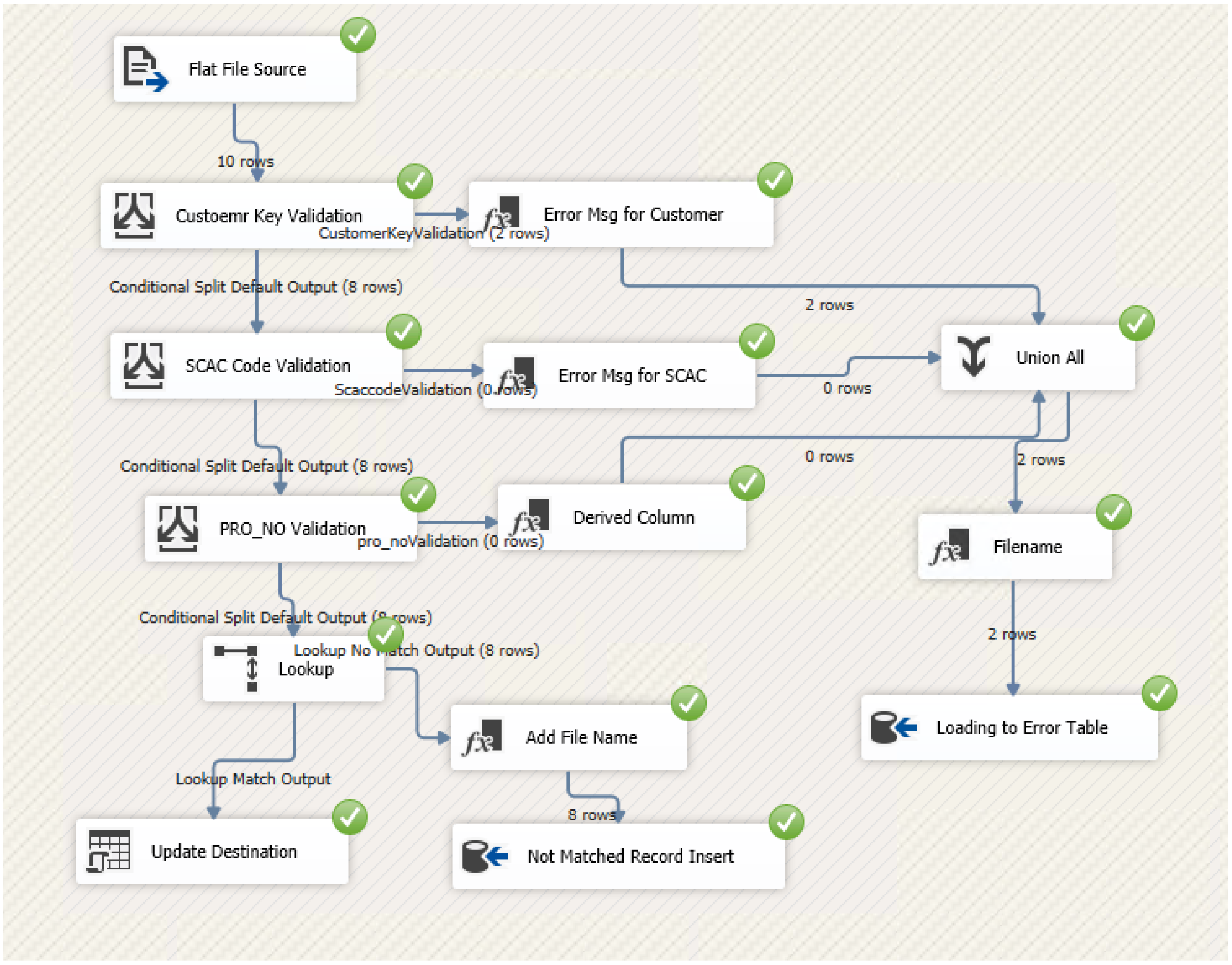
Let me run test run without pkg configurations (Now package is going to read variable and connection manager (connection string) values from package default values)

Control Flow task output:



Data Flow task output:

Verify below 10 records at the source level and two records are moved to error output and 8 records are inserted



Now let us work on Pkg configurations using SQL Server Tables (Important for interview point of view)

1) Source File Location :
   Source file location may change from development to QAT, QAT to prod …

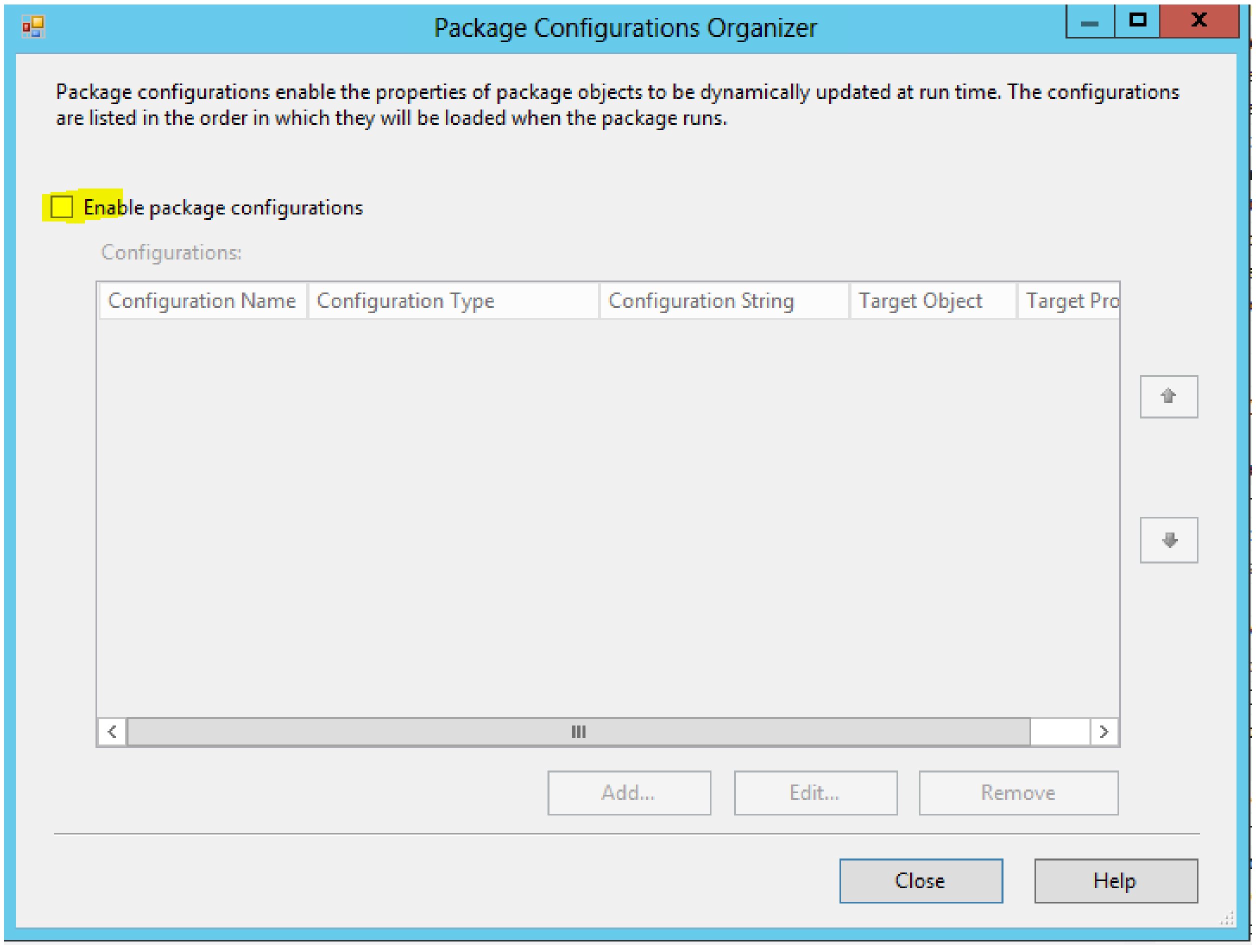
   Now if we deployed this package to Prod environment for the following scenario how the package is going to read from new location?

Source File Locations:

Development	QAT	PROD
C:ETL\PaymentSoruce \	C:ETL\Test\PaymentSource \	\\USATLFILES\ETL\IMPORT\PayementHistor y\

Now let us stored that variable value in the database table. By using the SQL Server table configurations.

Step 1: Enable the package configurations


Package Configurations Organizer

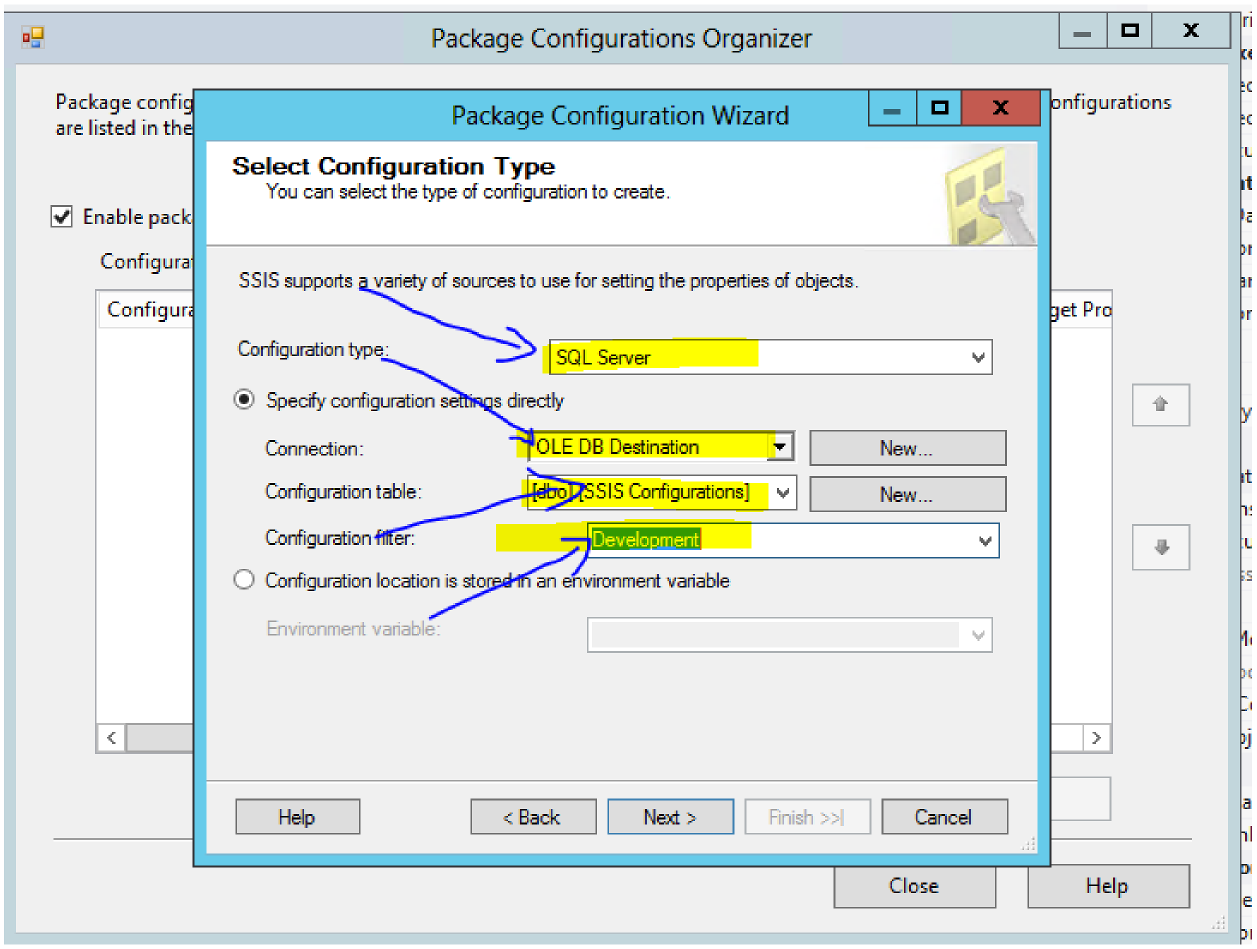
Package configurations enable the properties of package objects to be dynamically updated at run time. The configurations are listed in the order in which they will be loaded when the package runs.

☐ Enable package configurations

Configurations:

Configuration Name	Configuration Type	Configuration String	Target Object	Target Pro

Click on Enable package configurations
Then click on Add
Then select the following



Package Configurations Organizer

Package Configuration Wizard

Select Configuration Type
You can select the type of configuration to create.

SSIS supports a variety of sources to use for setting the properties of objects.

Configuration type: SQL Server

(●) Specify configuration settings directly

Connection: OLE DB Destination

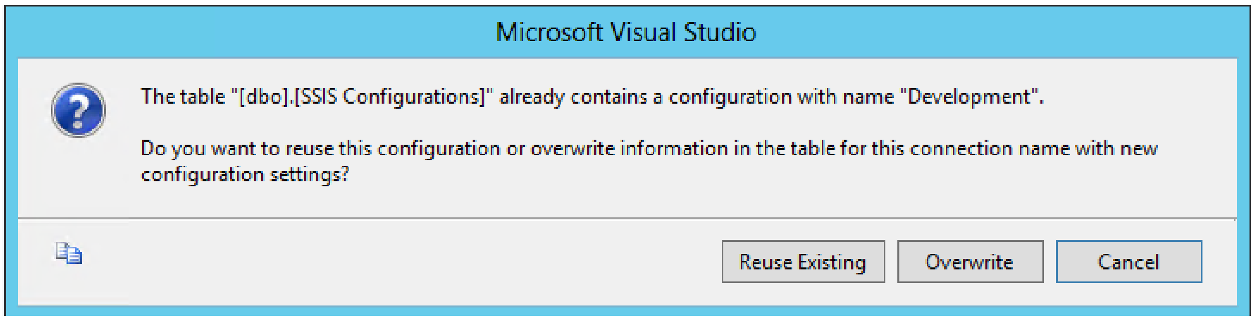
Configuration table: [dbo].[SSIS Configurations]

Configuration filter: Development

( ) Configuration location is stored in an environment variable

Environment variable:

If you got this message box click on Reuse existing

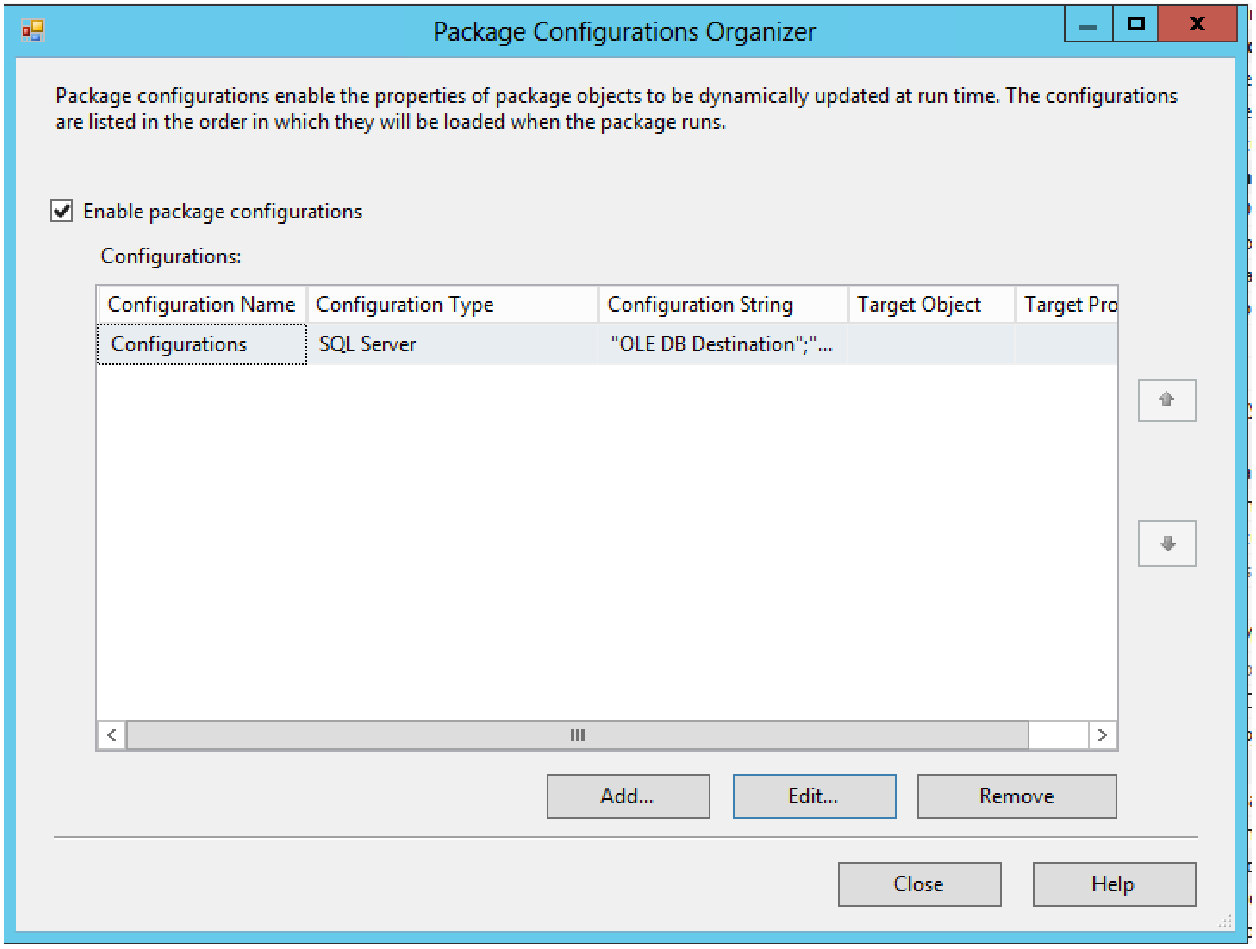


Microsoft Visual Studio

The table "[dbo].[SSIS Configurations]" already contains a configuration with name "Development".

Do you want to reuse this configuration or overwrite information in the table for this connection name with new configuration settings?

Then click on finish



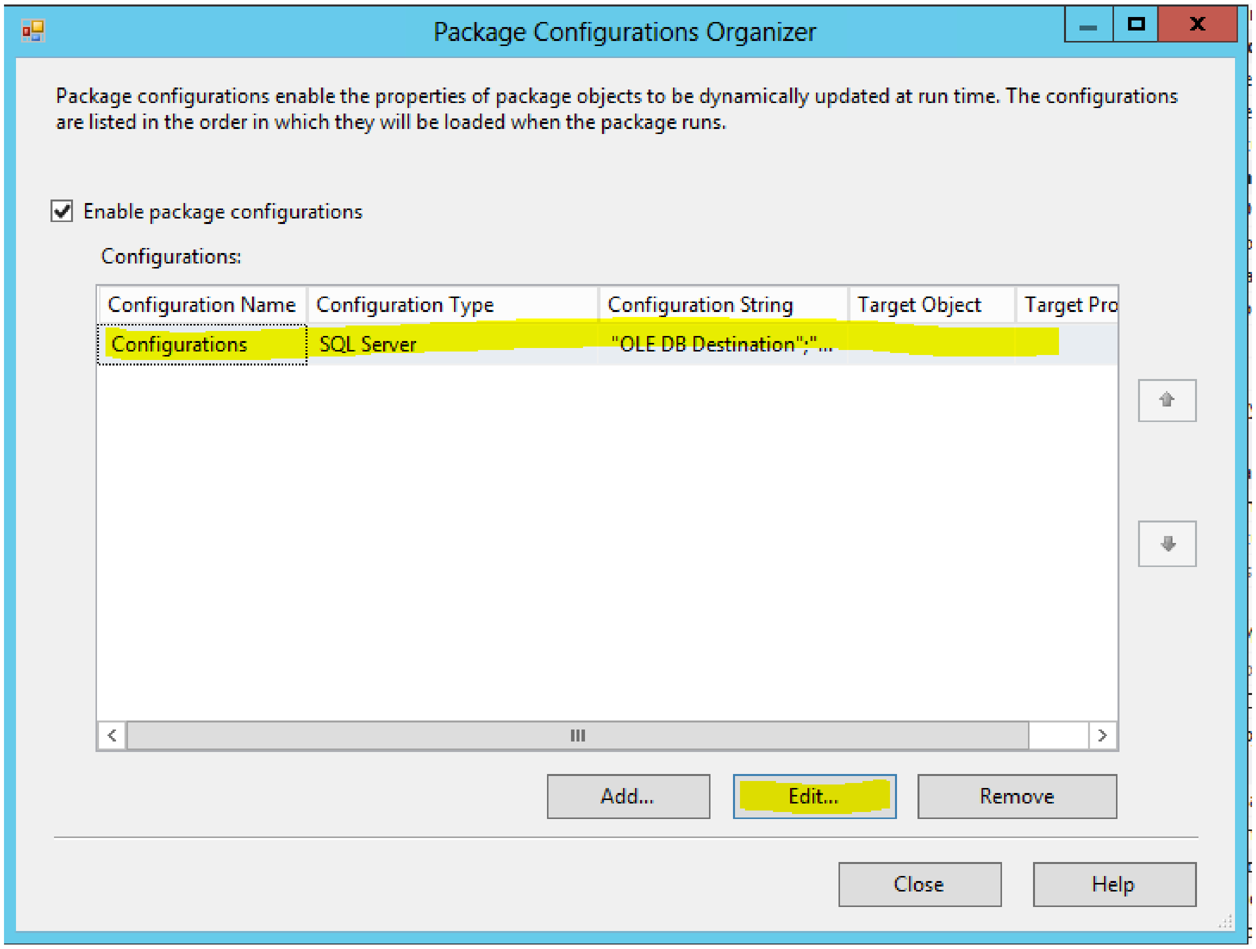
Package configurations enable the properties of package objects to be dynamically updated at run time. The configurations are listed in the order in which they will be loaded when the package runs.

☑ Enable package configurations

Configurations:

Configuration Name	Configuration Type	Configuration String	Target Object	Target Pro
Configurations	SQL Server	"OLE DB Destination";"...		

Now let us open configuration file to add source file location and other connection manager’s values to it



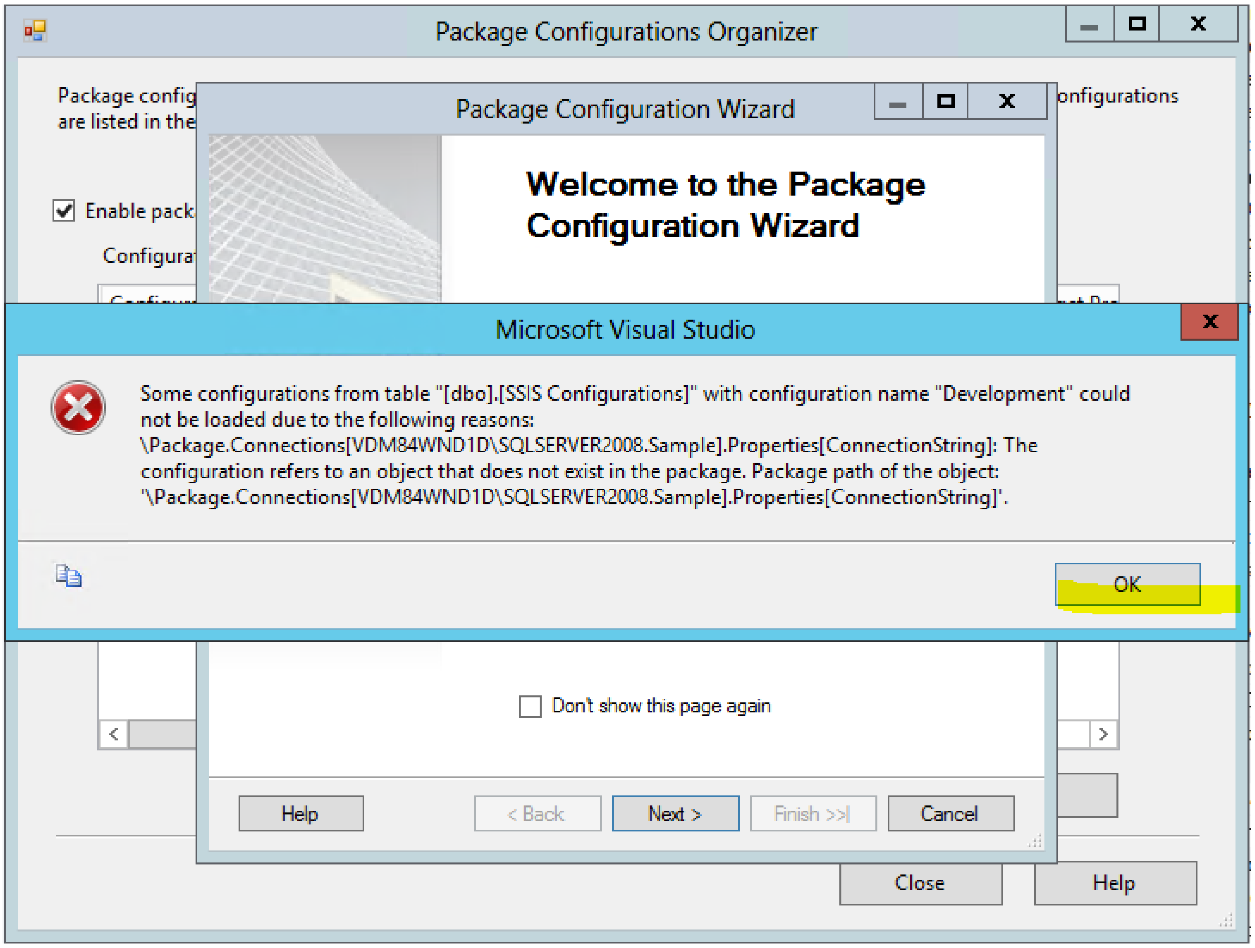
Package configurations enable the properties of package objects to be dynamically updated at run time. The configurations are listed in the order in which they will be loaded when the package runs.

☑ Enable package configurations

Configurations:

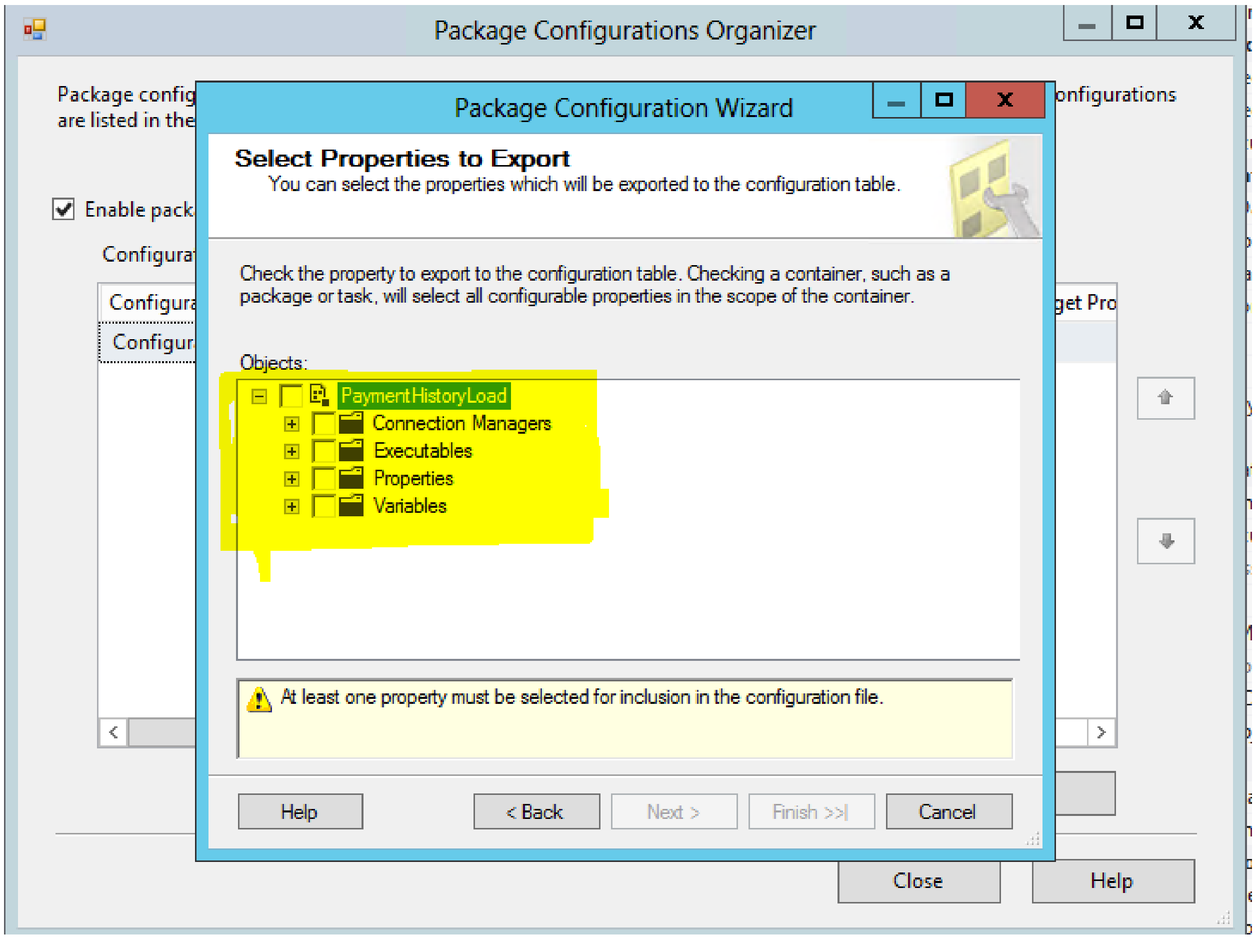
Configuration Name	Configuration Type	Configuration String	Target Object	Target Pro
Configurations	SQL Server	"OLE DB Destination";"...		

If you got this error click ok we will fix in the editor

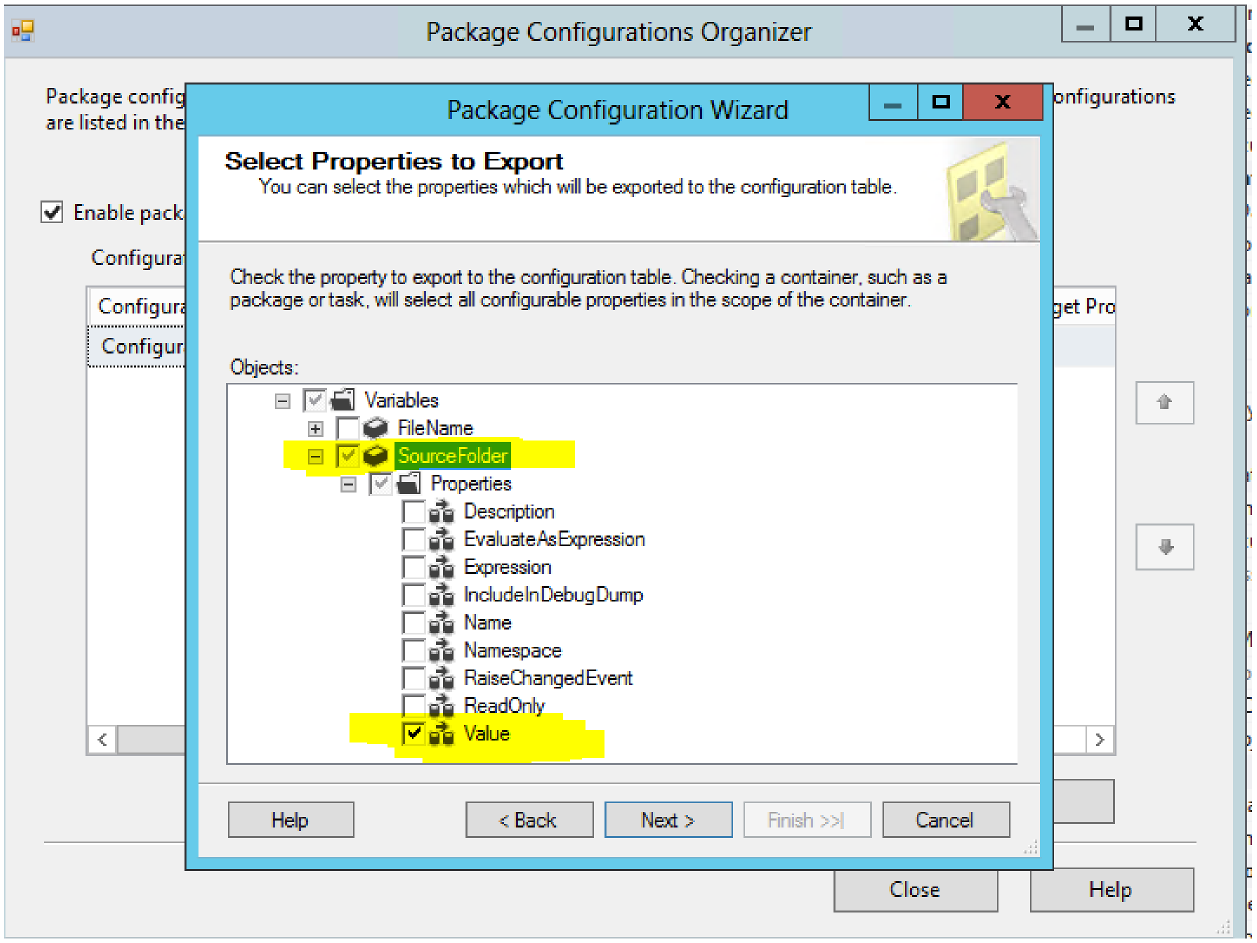


Some configurations from table "[dbo].[SSIS Configurations]" with configuration name "Development" could not be loaded due to the following reasons:
\Package.Connections[VDM84WND1D\SQLSERVER2008.Sample].Properties[ConnectionString]: The configuration refers to an object that does not exist in the package. Package path of the object: '\Package.Connections[VDM84WND1D\SQLSERVER2008.Sample].Properties[ConnectionString]'.

Now Click next here …


Package Configurations Organizer
Package config
are listed in the
Enable pack
Configura
Configura
Configur
onfigurations
get Pro
Package Configuration Wizard
Select Properties to Export
You can select the properties which will be exported to the configuration table.
Check the property to export to the configuration table. Checking a container, such as a package or task, will select all configurable properties in the scope of the container.
Objects:
PaymentHistoryLoad
Connection Managers
Executables
Properties
Variables
At least one property must be selected for inclusion in the configuration file.
Help
< Back
Next >
Finish >>|
Cancel
Close
Help

Step 2: Select variable Source Folder Location and Values


Package Configurations Organizer

Package Configuration Wizard

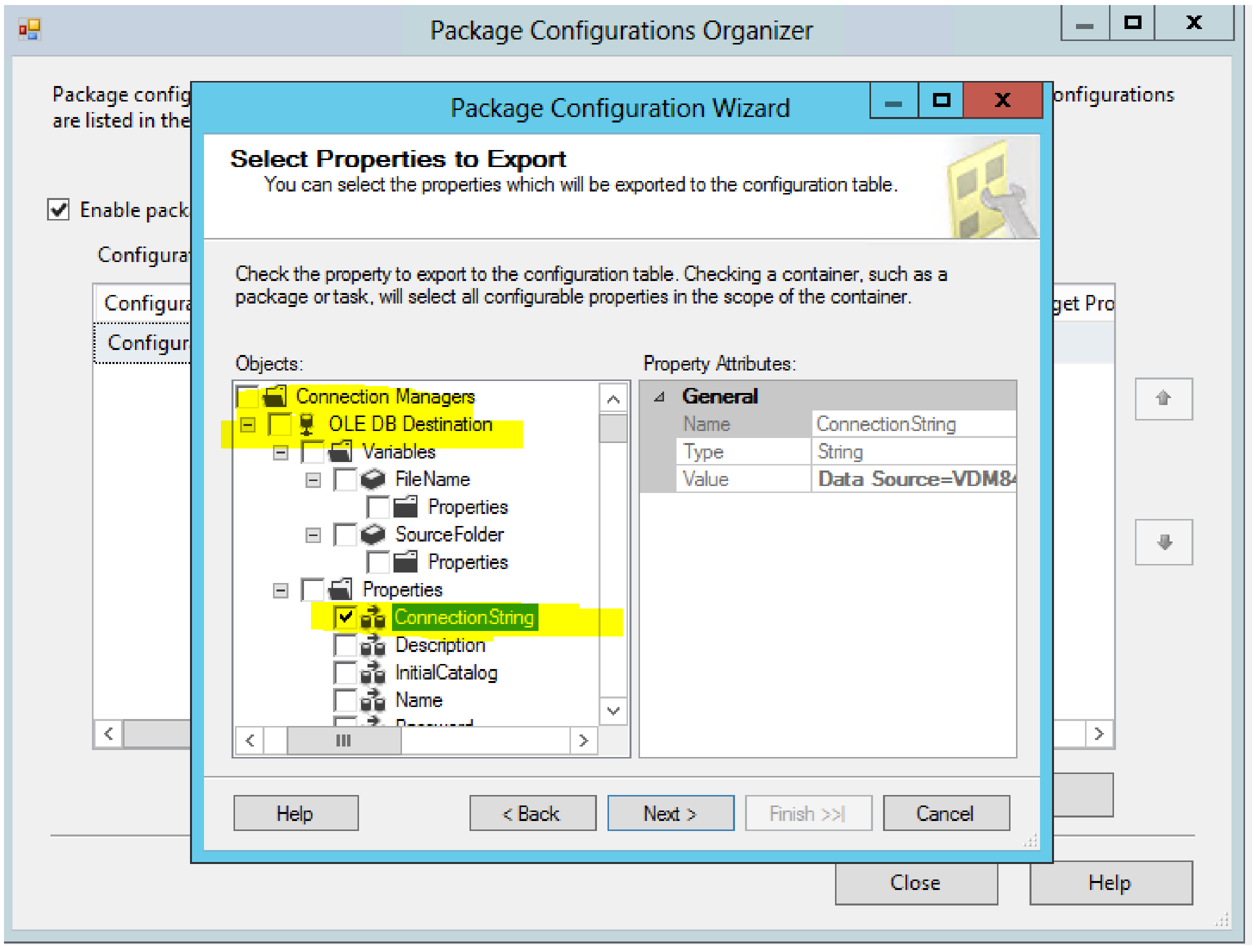
Select Properties to Export

You can select the properties which will be exported to the configuration table.

Check the property to export to the configuration table. Checking a container, such as a package or task, will select all configurable properties in the scope of the container.

Objects:

```
⊟ ☑ Variables
    ⊞ ☐ FileName
    ⊟ ☑ SourceFolder
        ⊟ ☑ Properties
            ☐ Description
            ☐ EvaluateAsExpression
            ☐ Expression
            ☐ IncludeInDebugDump
            ☐ Name
            ☐ Namespace
            ☐ RaiseChangedEvent
            ☐ ReadOnly
            ☑ Value
```



Select Properties to Export
You can select the properties which will be exported to the configuration table.

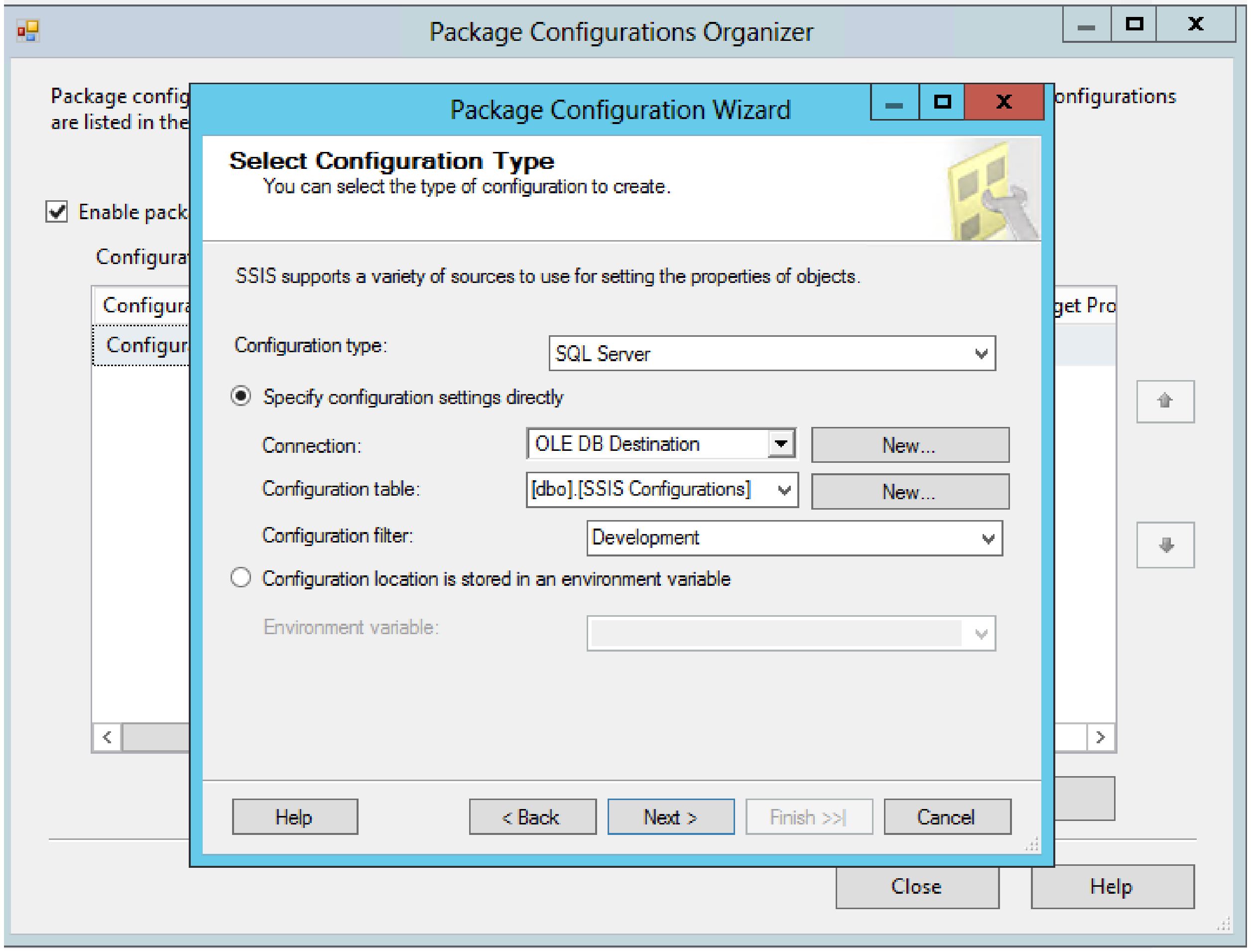
Check the property to export to the configuration table. Checking a container, such as a package or task, will select all configurable properties in the scope of the container.

Objects:

```
[ ] Connection Managers
  [ ] OLE DB Destination
    [ ] Variables
      [ ] FileName
        [ ] Properties
      [ ] SourceFolder
        [ ] Properties
    [ ] Properties
      [✓] ConnectionString
      [ ] Description
      [ ] InitialCatalog
      [ ] Name
```

Property Attributes:

General	
Name	ConnectionString
Type	String
Value	Data Source=VDM84



Package Configuration Wizard

Select Configuration Type

You can select the type of configuration to create.

SSIS supports a variety of sources to use for setting the properties of objects.

Configuration type: SQL Server

(●) Specify configuration settings directly

Connection: OLE DB Destination

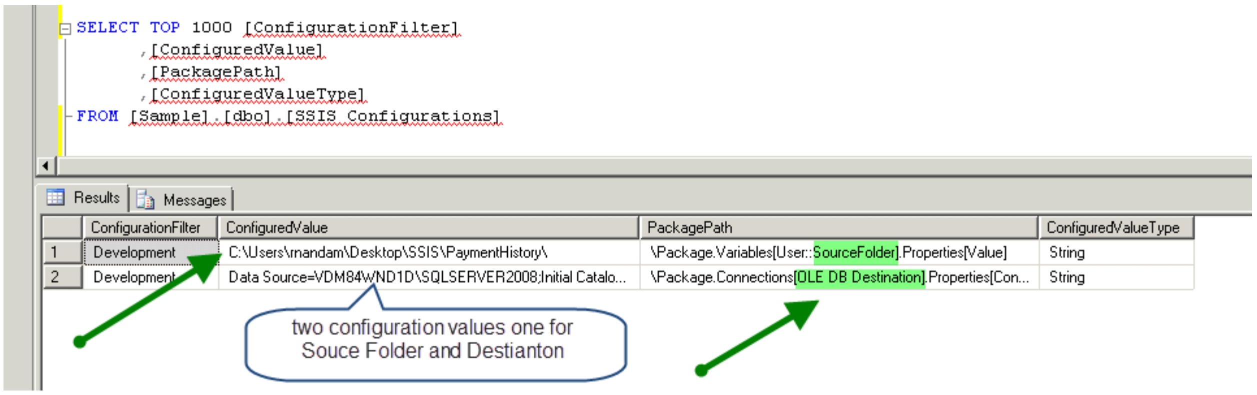
Configuration table: [dbo].[SSIS Configurations]

Configuration filter: Development

( ) Configuration location is stored in an environment variable

Environment variable:

After selecting those configurations, let me see values in configuration tables



```
SELECT TOP 1000 [ConfigurationFilter]
      ,[ConfiguredValue]
      ,[PackagePath]
      ,[ConfiguredValueType]
FROM [Sample].[dbo].[SSIS Configurations]
```

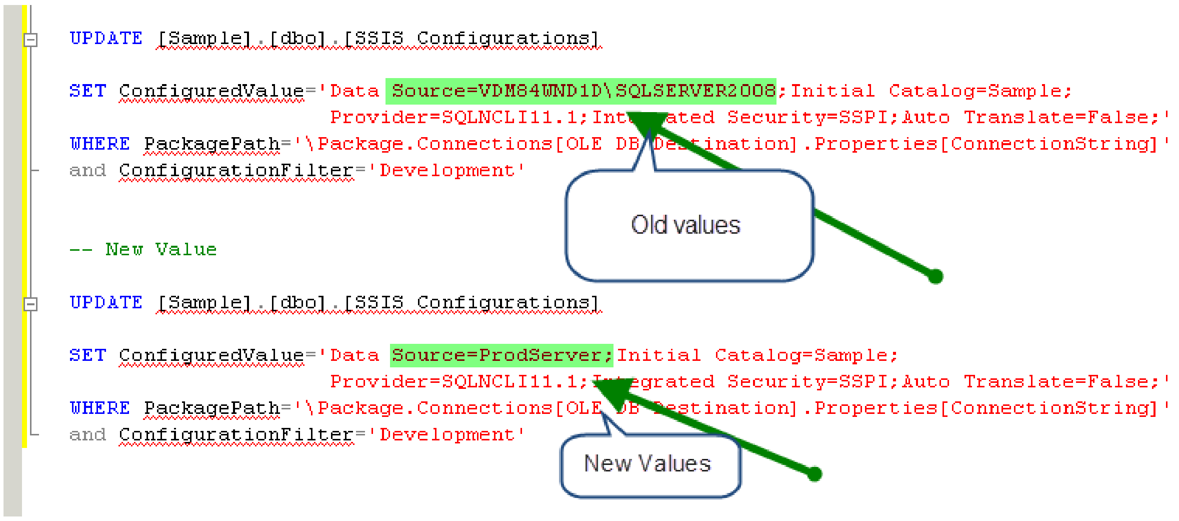
	ConfigurationFilter	ConfiguredValue	PackagePath	ConfiguredValueType
1	Development	C:\Users\mandam\Desktop\SSIS\PaymentHistory\	\Package.Variables[User::SourceFolder].Properties[Value]	String
2	Development	Data Source=VDM84WND1D\SQLSERVER2008;Initial Catalo...	\Package.Connections[OLE DB Destination].Properties[Con...	String

Test the Configurations

Let us assume we moved our package to the Production server

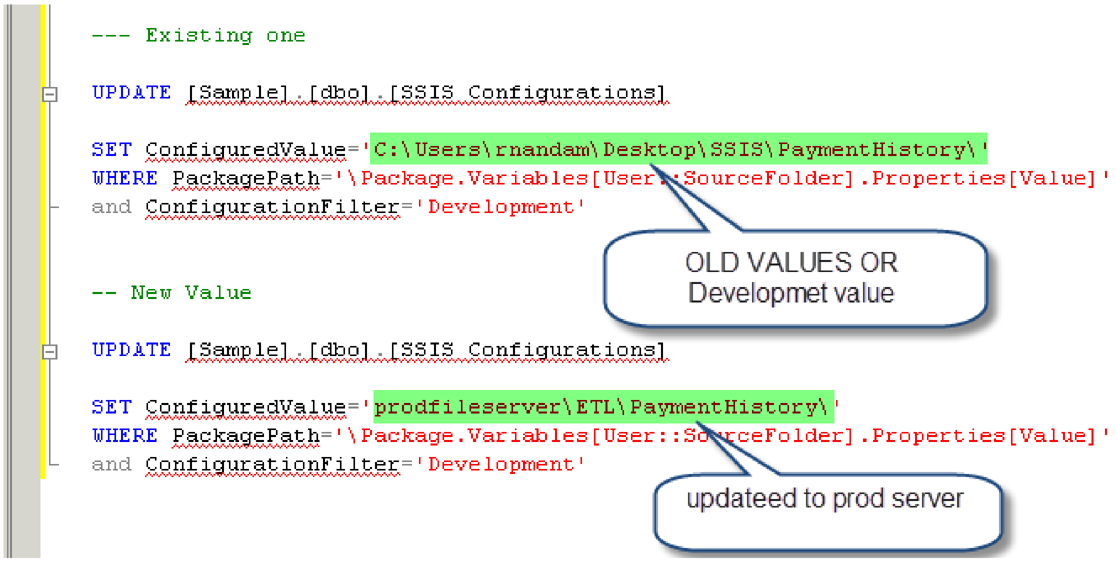
Based on the production server name we will update OLE DB Destination Connection in the configuration table

Example statement here



```
UPDATE [Sample].[dbo].[SSIS Configurations]

SET ConfiguredValue='Data Source=VDM84WND1D\SQLSERVER2008;Initial Catalog=Sample;
                     Provider=SQLNCLI11.1;Integrated Security=SSPI;Auto Translate=False;'
WHERE PackagePath='\Package.Connections[OLE DB Destination].Properties[ConnectionString]'
and ConfigurationFilter='Development'


-- New Value

UPDATE [Sample].[dbo].[SSIS Configurations]

SET ConfiguredValue='Data Source=ProdServer;Initial Catalog=Sample;
                     Provider=SQLNCLI11.1;Integrated Security=SSPI;Auto Translate=False;'
WHERE PackagePath='\Package.Connections[OLE DB Destination].Properties[ConnectionString]'
and ConfigurationFilter='Development'
```

In the same way we do for the other connection’s or variable values in the Configuration table, so that package is going to read those values at run time



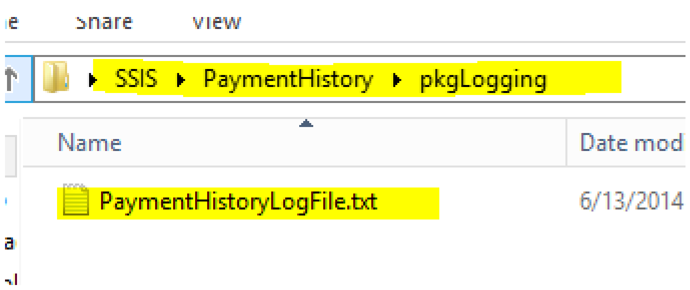
```
--- Existing one

UPDATE [Sample].[dbo].[SSIS Configurations]

SET ConfiguredValue='C:\Users\rnandam\Desktop\SSIS\PaymentHistory\'
WHERE PackagePath='\Package.Variables[User::SourceFolder].Properties[Value]'
and ConfigurationFilter='Development'

-- New Value

UPDATE [Sample].[dbo].[SSIS Configurations]

SET ConfiguredValue='prodfileserver\ETL\PaymentHistory\'
WHERE PackagePath='\Package.Variables[User::SourceFolder].Properties[Value]'
and ConfigurationFilter='Development'
```

Now let us work on the Package Logging

- Package Logging using the Flat File log file
- Package logging using the SQL Server Table

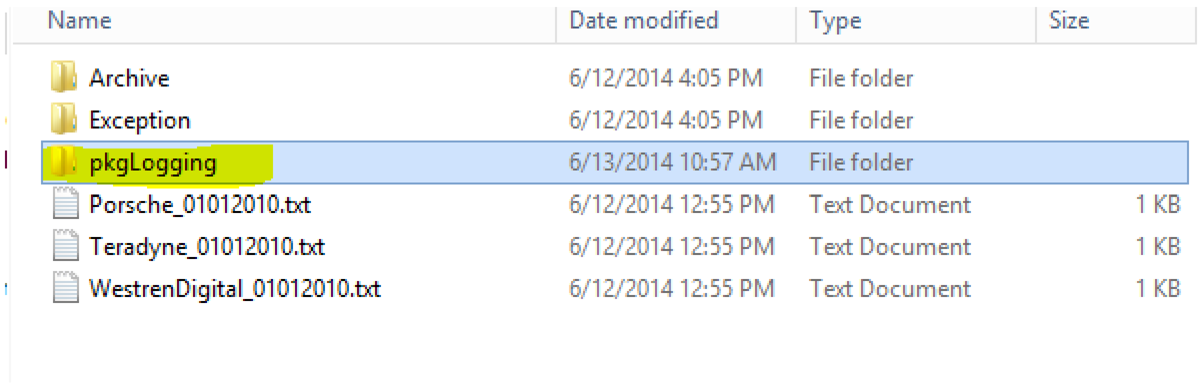
Logging configuration using Flat File

Step 1: Create the Log file (PayementHistoryLogFile.txt)



PaymentHistoryLogFile.txt 6/13/2014

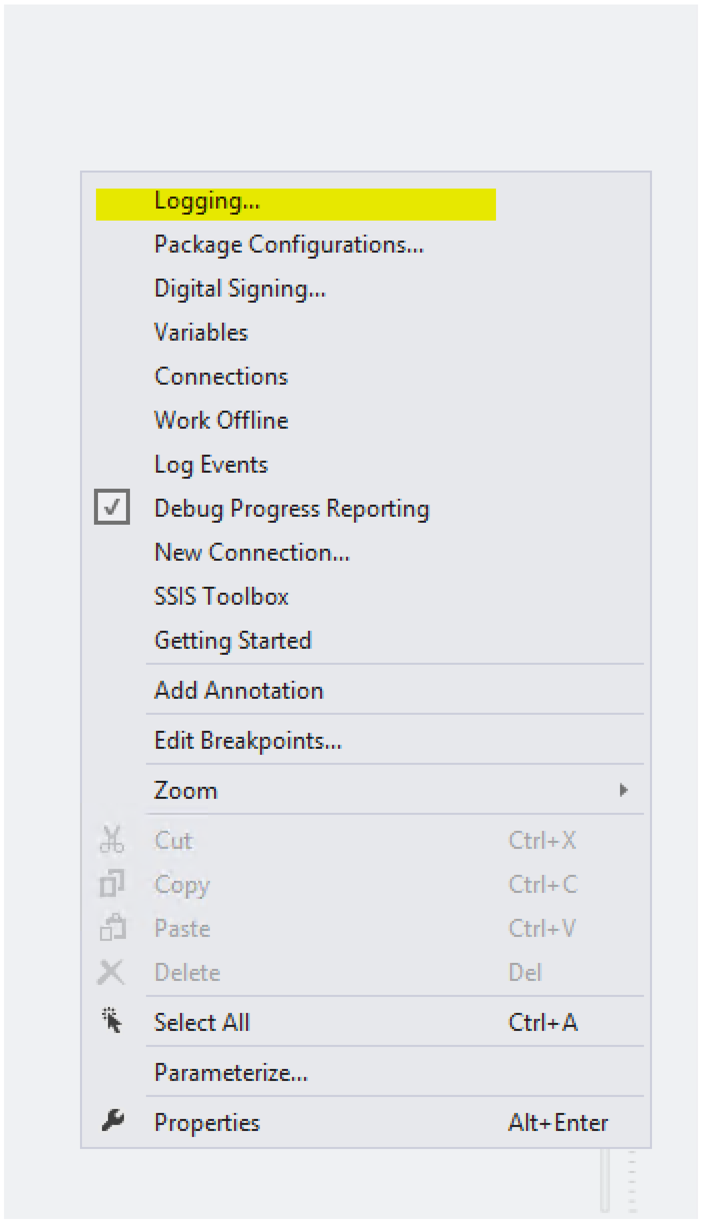
Folder Structure



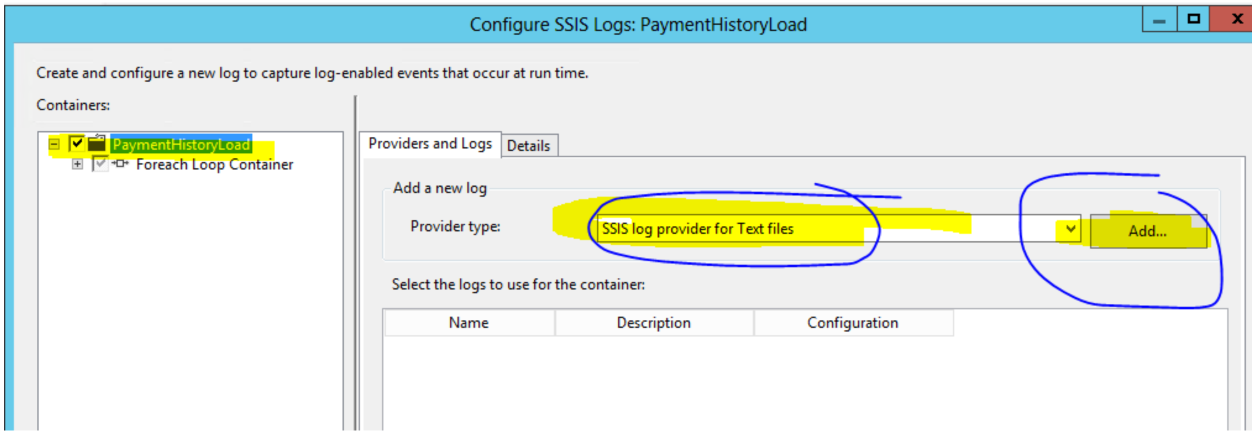
Name	Date modified	Type	Size
Archive	6/12/2014 4:05 PM	File folder	
Exception	6/12/2014 4:05 PM	File folder	
pkgLogging	6/13/2014 10:57 AM	File folder	
Porsche_01012010.txt	6/12/2014 12:55 PM	Text Document	1 KB
Teradyne_01012010.txt	6/12/2014 12:55 PM	Text Document	1 KB
WestrenDigital_01012010.txt	6/12/2014 12:55 PM	Text Document	1 KB

Step 2: Add Logging to package

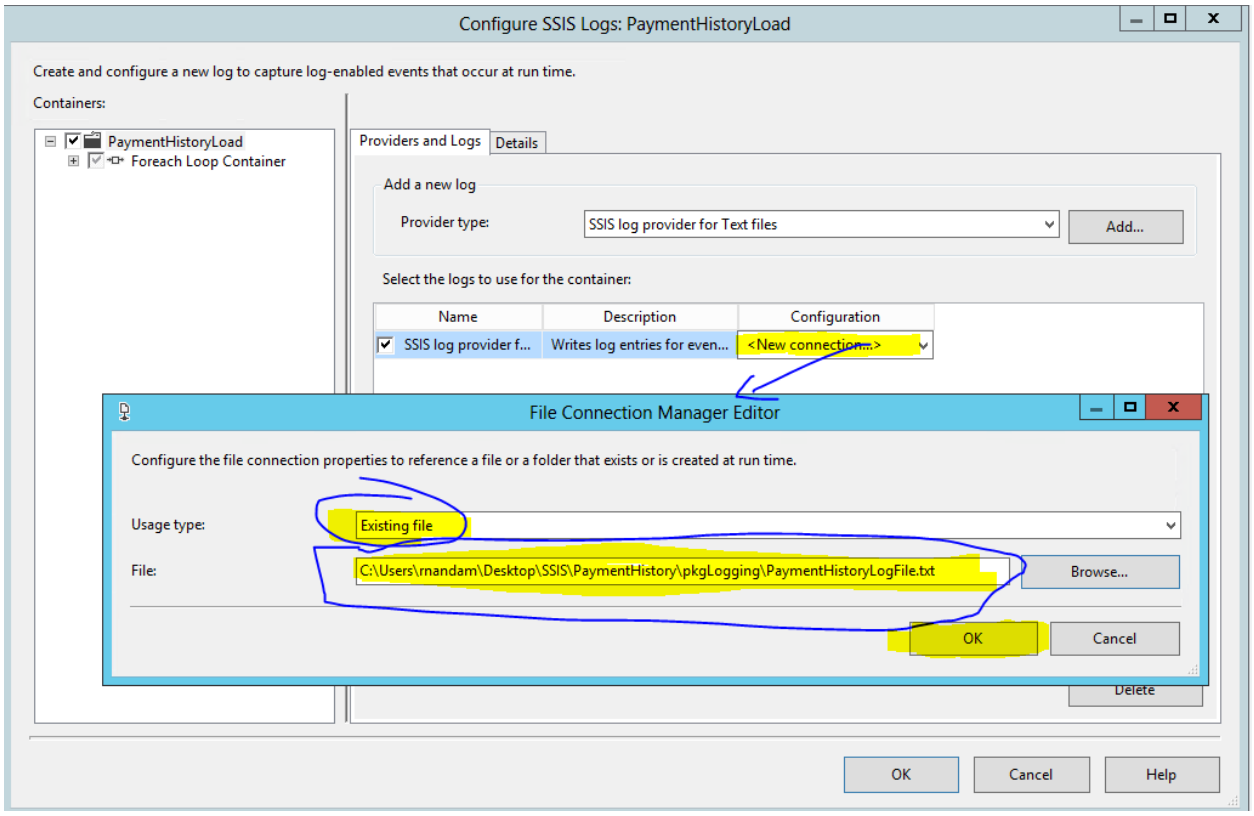
Right Click on the control flow design area and click on logging



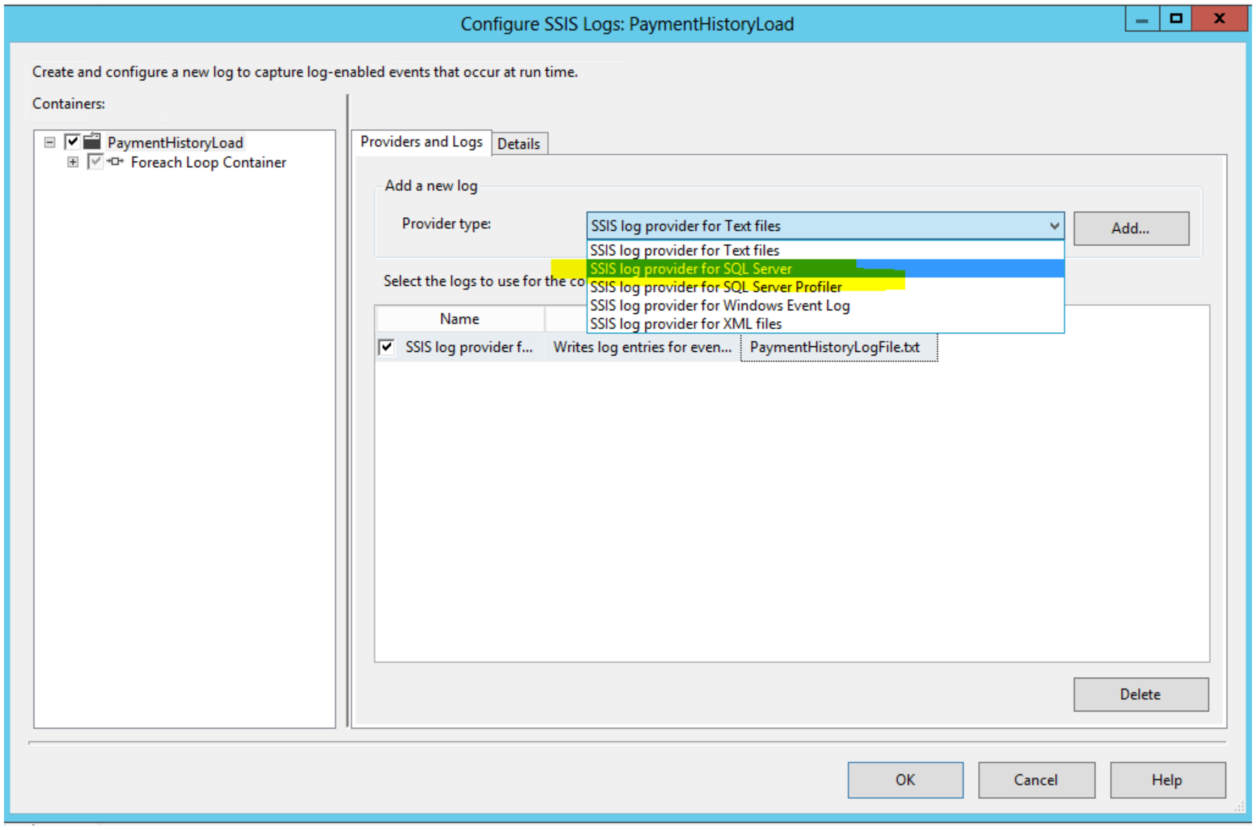
In the logging wizard select log provider type and add it


Configure SSIS Logs: PaymentHistoryLoad
Create and configure a new log to capture log-enabled events that occur at run time.
Containers:
PaymentHistoryLoad
Foreach Loop Container
Providers and Logs
Details
Add a new log
Provider type:
SSIS log provider for Text files
Add...
Select the logs to use for the container:
Name
Description
Configuration

Click on new connection then select existing file and select the log file from the log directory


Configure SSIS Logs: PaymentHistoryLoad
Create and configure a new log to capture log-enabled events that occur at run time.
Containers:
PaymentHistoryLoad
Foreach Loop Container
Providers and Logs
Details
Add a new log
Provider type:
SSIS log provider for Text files
Add...
Select the logs to use for the container:
Name
Description
Configuration
SSIS log provider f...
Writes log entries for even...
<New connection...>
File Connection Manager Editor
Configure the file connection properties to reference a file or a folder that exists or is created at run time.
Usage type:
Existing file
File:
C:\Users\rnandam\Desktop\SSIS\PaymentHistory\pkgLogging\PaymentHistoryLogFile.txt
Browse...
OK
Cancel
Delete
OK
Cancel
Help

Then Add SQL Server log type to the package



Configure SSIS Logs: PaymentHistoryLoad

Create and configure a new log to capture log-enabled events that occur at run time.

Containers:

- PaymentHistoryLoad
  - Foreach Loop Container

Providers and Logs | Details

Add a new log

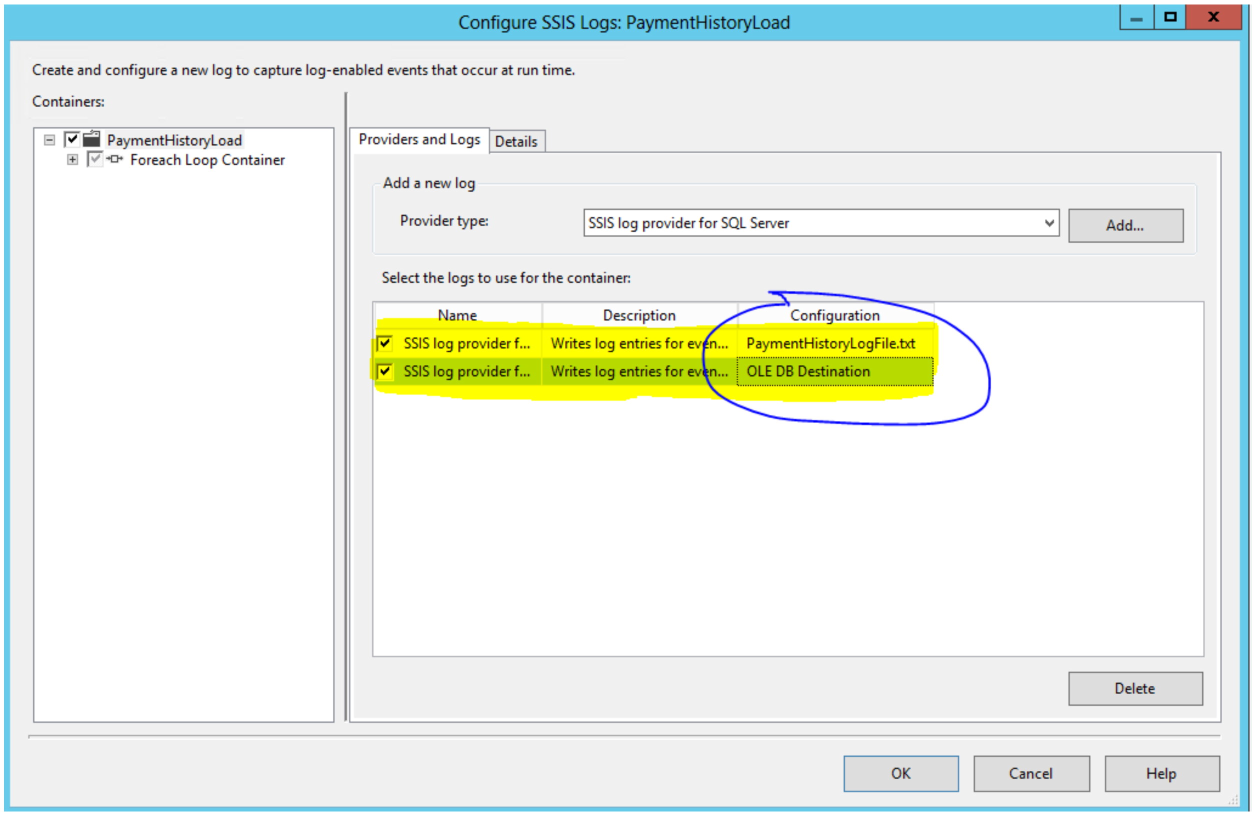
Provider type: SSIS log provider for Text files

- SSIS log provider for Text files
- SSIS log provider for SQL Server
- SSIS log provider for SQL Server Profiler
- SSIS log provider for Windows Event Log
- SSIS log provider for XML files

Select the logs to use for the co

Name		
✓ SSIS log provider f...	Writes log entries for even...	PaymentHistoryLogFile.txt

And select Configuration as OLEDB Destination


Configure SSIS Logs: PaymentHistoryLoad

Create and configure a new log to capture log-enabled events that occur at run time.

Containers:

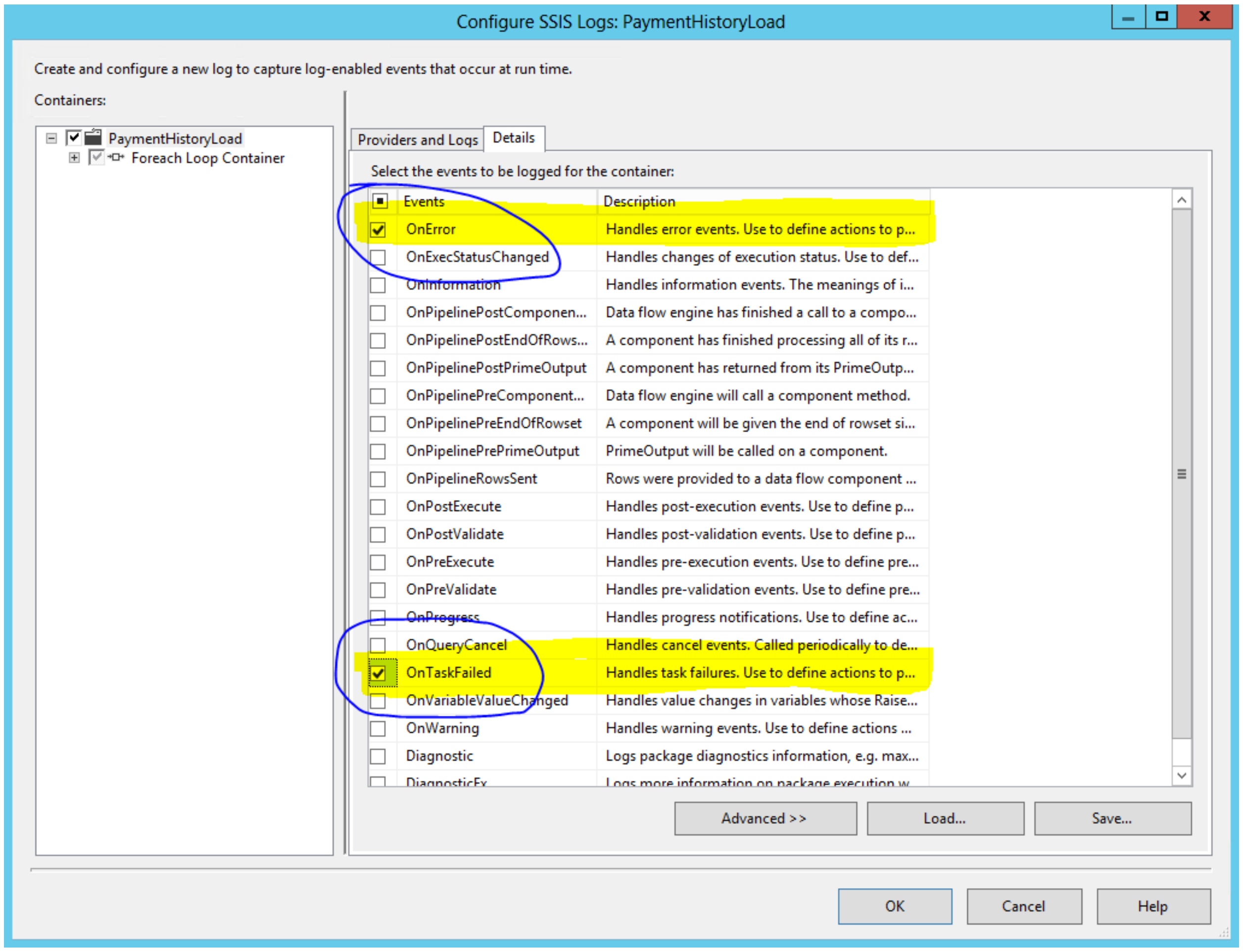
- [✓] PaymentHistoryLoad
  - [✓] Foreach Loop Container

Provider type: SSIS log provider for SQL Server

Select the logs to use for the container:

	Name	Description	Configuration
✓	SSIS log provider f...	Writes log entries for even...	PaymentHistoryLogFile.txt
✓	SSIS log provider f...	Writes log entries for even...	OLE DB Destination

Step 3: Now Select the Details for events onError and OnTaskFailed (You can select more based on your company or environment need)
Then Click OK



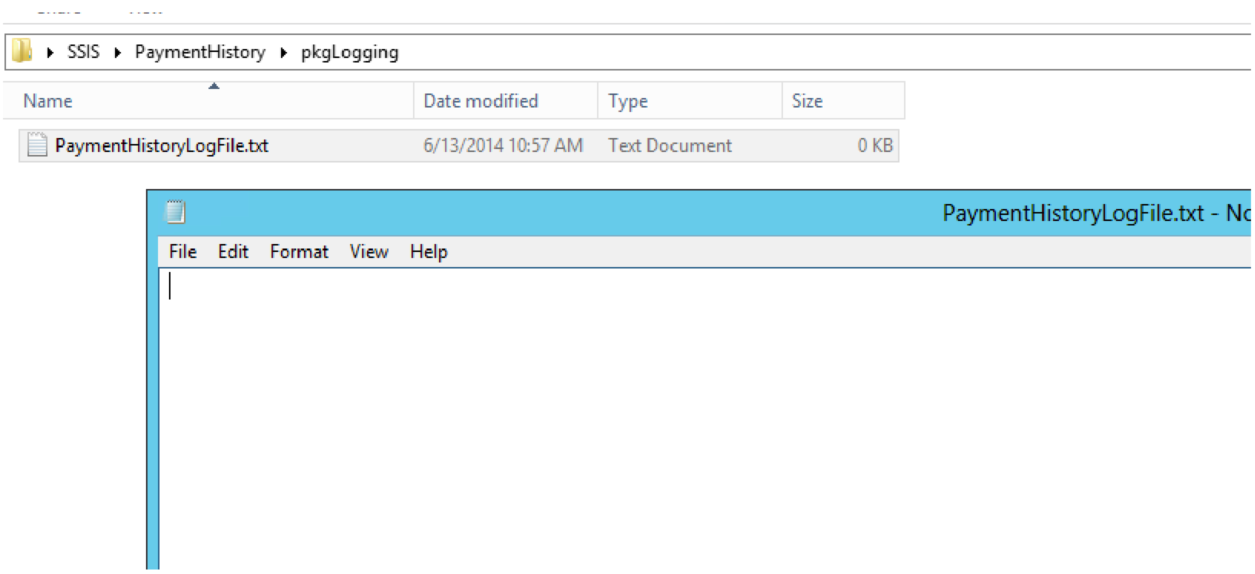
Create and configure a new log to capture log-enabled events that occur at run time.

Select the events to be logged for the container:

	Events	Description
✓	OnError	Handles error events. Use to define actions to p...
	OnExecStatusChanged	Handles changes of execution status. Use to def...
	OnInformation	Handles information events. The meanings of i...
	OnPipelinePostComponen...	Data flow engine has finished a call to a compo...
	OnPipelinePostEndOfRows...	A component has finished processing all of its r...
	OnPipelinePostPrimeOutput	A component has returned from its PrimeOutp...
	OnPipelinePreComponent...	Data flow engine will call a component method.
	OnPipelinePreEndOfRowset	A component will be given the end of rowset si...
	OnPipelinePrePrimeOutput	PrimeOutput will be called on a component.
	OnPipelineRowsSent	Rows were provided to a data flow component ...
	OnPostExecute	Handles post-execution events. Use to define p...
	OnPostValidate	Handles post-validation events. Use to define p...
	OnPreExecute	Handles pre-execution events. Use to define pre...
	OnPreValidate	Handles pre-validation events. Use to define pre...
	OnProgress	Handles progress notifications. Use to define ac...
	OnQueryCancel	Handles cancel events. Called periodically to de...
✓	OnTaskFailed	Handles task failures. Use to define actions to p...
	OnVariableValueChanged	Handles value changes in variables whose Raise...
	OnWarning	Handles warning events. Use to define actions ...
	Diagnostic	Logs package diagnostics information, e.g. max...
	DiagnosticEx	Logs more information on package execution w...

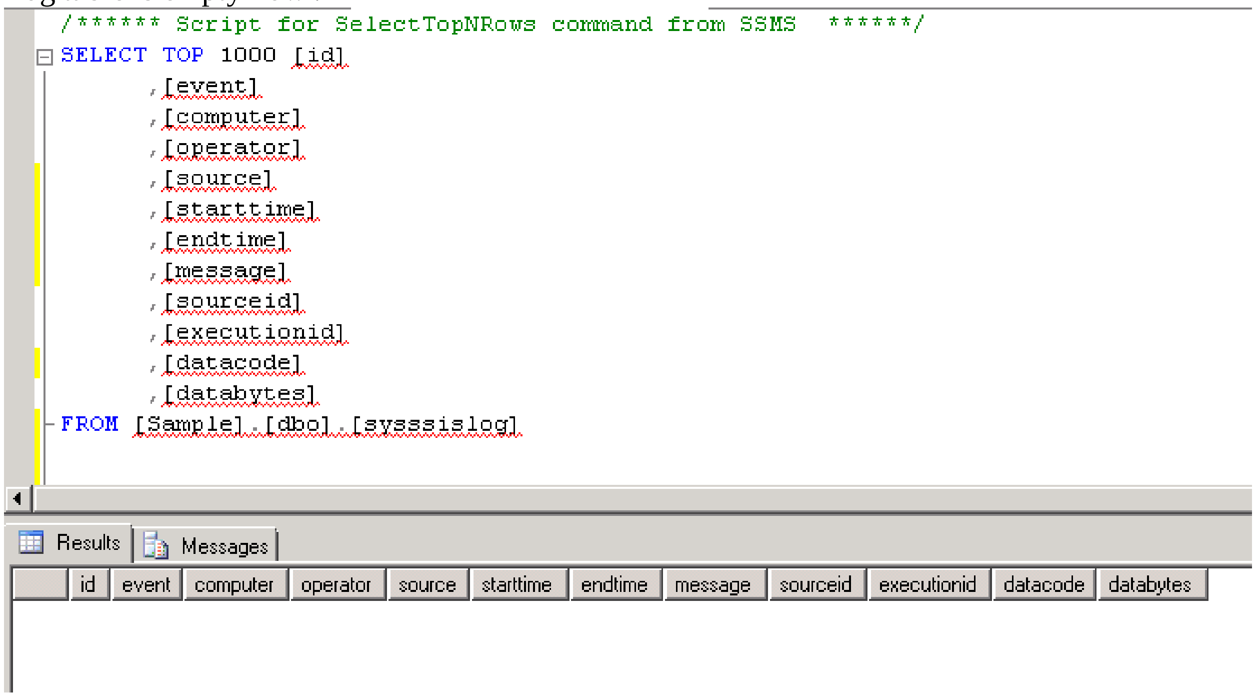
Now let us verify both log file and log table before running the package :

Log file is Empty



Name	Date modified	Type	Size
PaymentHistoryLogFile.txt	6/13/2014 10:57 AM	Text Document	0 KB

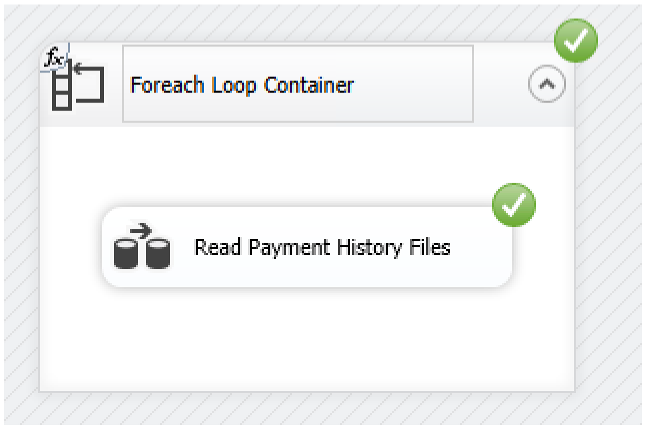
Log table is empty now :



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [id]
      ,[event]
      ,[computer]
      ,[operator]
      ,[source]
      ,[starttime]
      ,[endtime]
      ,[message]
      ,[sourceid]
      ,[executionid]
      ,[datacode]
      ,[databytes]
  FROM [Sample].[dbo].[sysssislog]
```

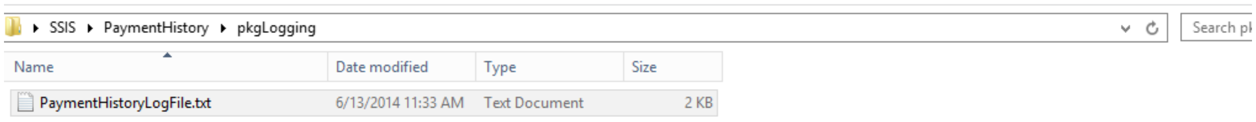
	id	event	computer	operator	source	starttime	endtime	message	sourceid	executionid	datacode	databytes

Let us run the package to test the log File and Log table:

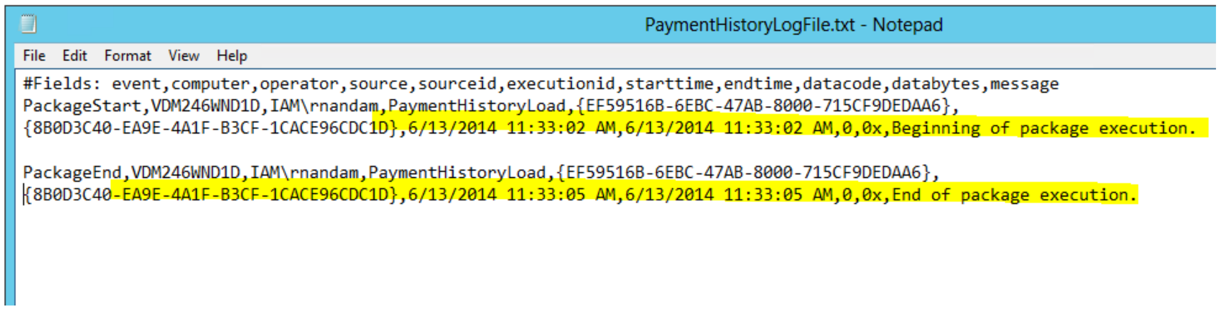


Let us verify the package logging now,

I have just package start and end time, if any time package got failed it will log that info

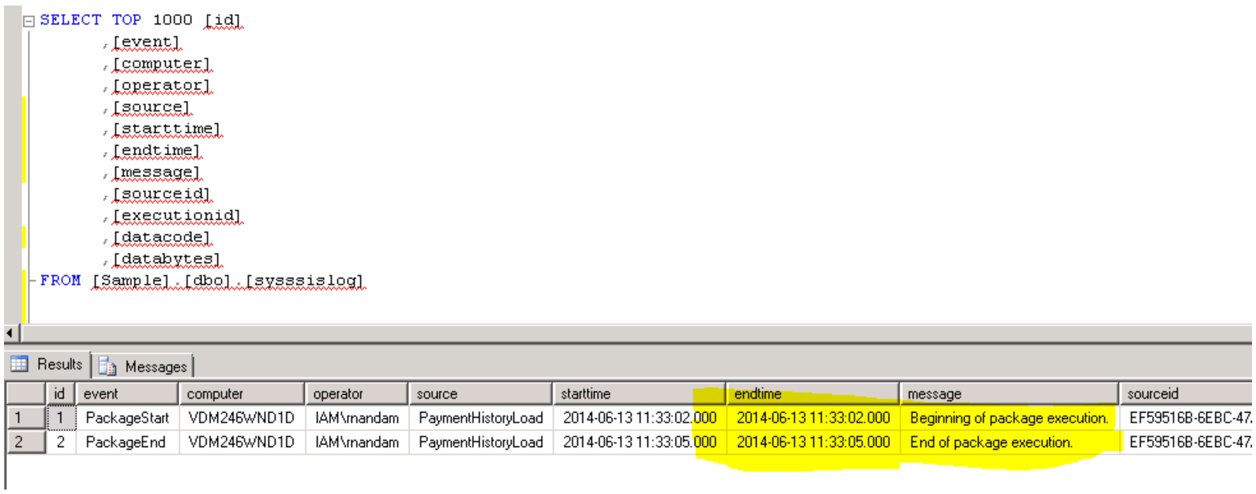


Name	Date modified	Type	Size
PaymentHistoryLogFile.txt	6/13/2014 11:33 AM	Text Document	2 KB


PaymentHistoryLogFile.txt - Notepad
File Edit Format View Help

```
#Fields: event,computer,operator,source,sourceid,executionid,starttime,endtime,datacode,databytes,message
PackageStart,VDM246WND1D,IAM\rnandam,PaymentHistoryLoad,{EF59516B-6EBC-47AB-8000-715CF9DEDAA6},
{8B0D3C40-EA9E-4A1F-B3CF-1CACE96CDC1D},6/13/2014 11:33:02 AM,6/13/2014 11:33:02 AM,0,0x,Beginning of package execution.

PackageEnd,VDM246WND1D,IAM\rnandam,PaymentHistoryLoad,{EF59516B-6EBC-47AB-8000-715CF9DEDAA6},
{8B0D3C40-EA9E-4A1F-B3CF-1CACE96CDC1D},6/13/2014 11:33:05 AM,6/13/2014 11:33:05 AM,0,0x,End of package execution.
```

Verifying the Log SQL Server Table



```
SELECT TOP 1000 [id]
      ,[event]
      ,[computer]
      ,[operator]
      ,[source]
      ,[starttime]
      ,[endtime]
      ,[message]
      ,[sourceid]
      ,[executionid]
      ,[datacode]
      ,[databytes]
FROM [Sample].[dbo].[sysssislog]
```

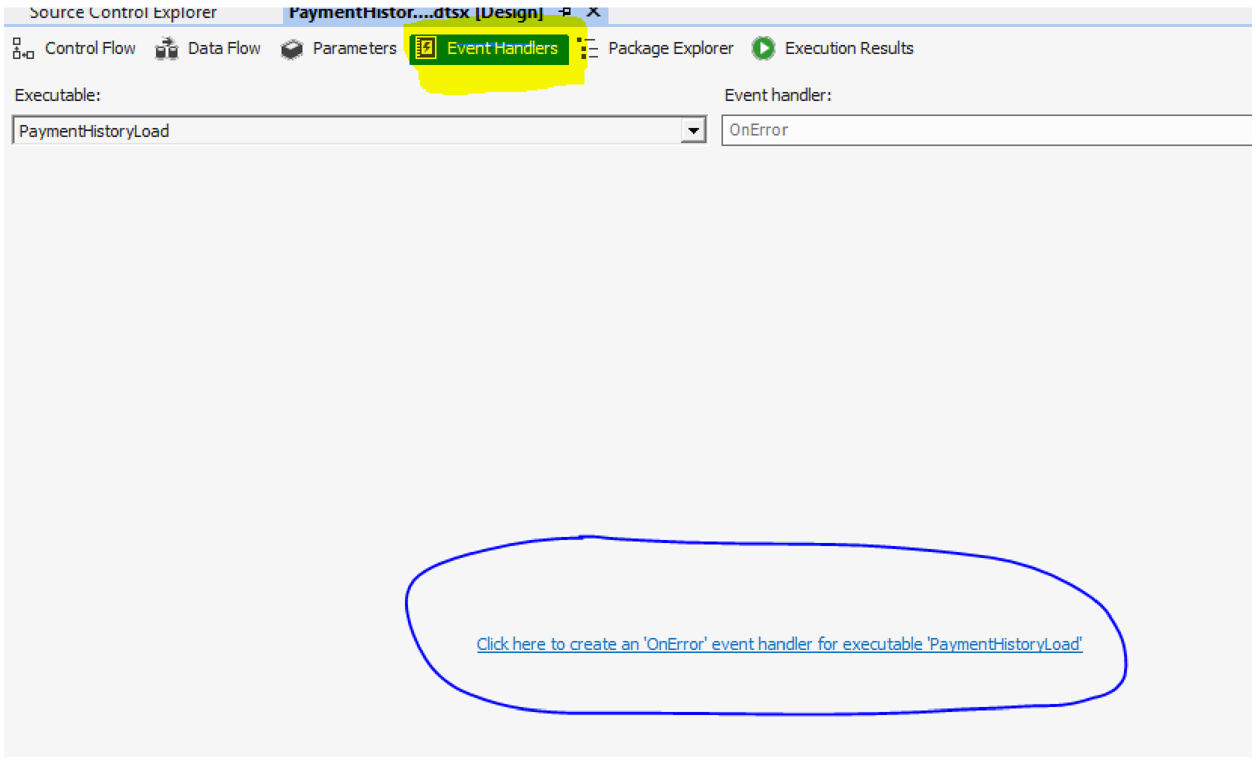
	id	event	computer	operator	source	starttime	endtime	message	sourceid
1	1	PackageStart	VDM246WND1D	IAM\rnandam	PaymentHistoryLoad	2014-06-13 11:33:02.000	2014-06-13 11:33:02.000	Beginning of package execution.	EF59516B-6EBC-47
2	2	PackageEnd	VDM246WND1D	IAM\rnandam	PaymentHistoryLoad	2014-06-13 11:33:05.000	2014-06-13 11:33:05.000	End of package execution.	EF59516B-6EBC-47

Implementing Event Handling to the Package.

Make sure you have custom event handling table. If not please create this table to log custom events from the package.

```
--- Create Custom Log Table for Event Handling

CREATE TABLE [dbo].[SSIS_Event_Log](
          [ID] [int] IDENTITY(1,1) NOT NULL,
          [PackageID] [uniqueidentifier] NULL,
          [Error] [nvarchar](max) NULL,
          [Source] [nvarchar](100) NULL,
          [PackageName] [nvarchar](100) NULL
          )
```

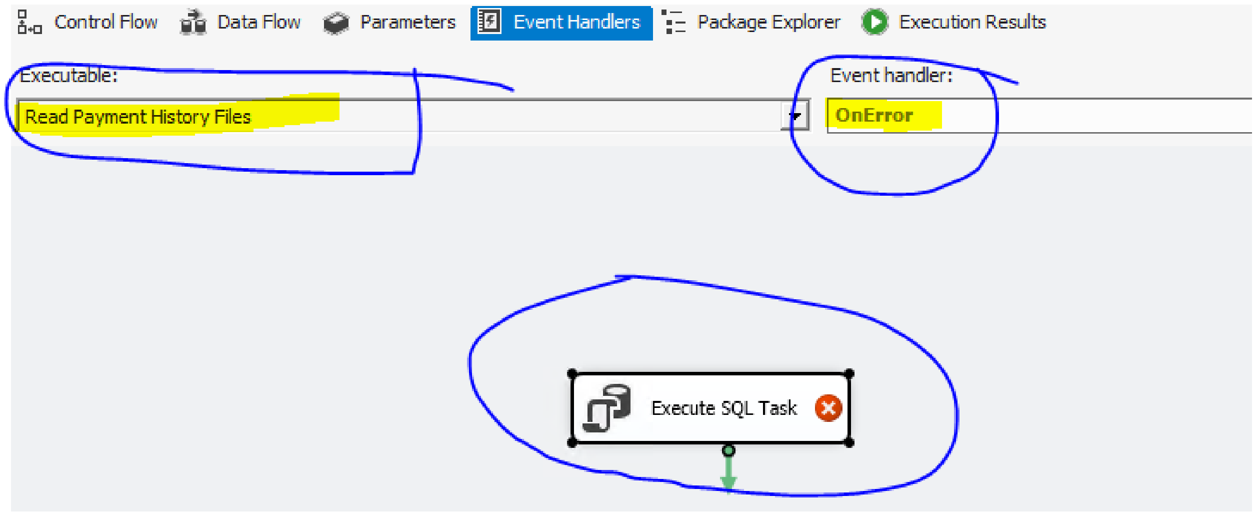
Step 1: Create Event handler for the on error,
If data flow task got failed, then we need to capture the error message and send email notification to the user



Executable: PaymentHistoryLoad
Event handler: OnError

Click here to create an 'OnError' event handler for executable 'PaymentHistoryLoad'

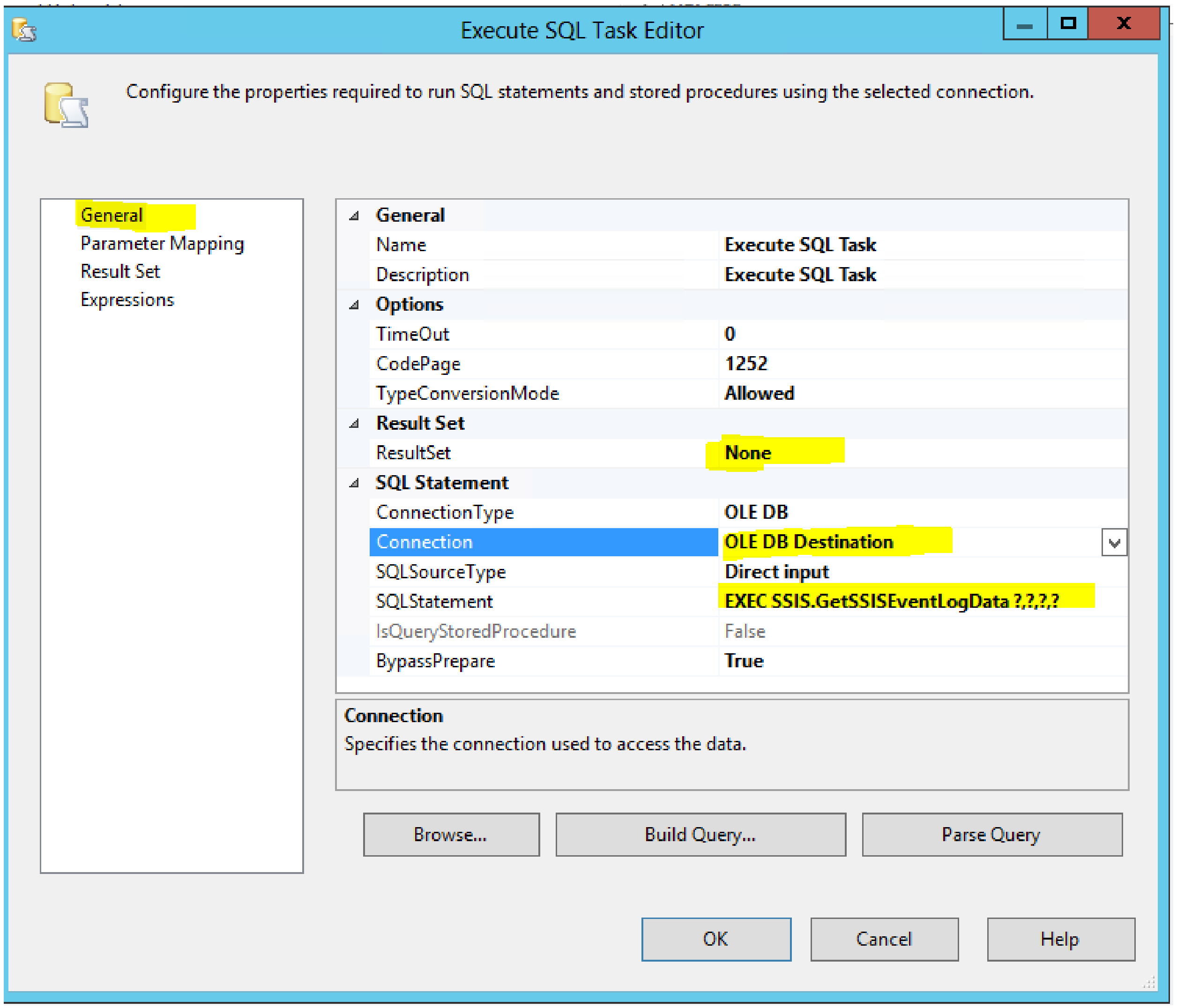
Let us drag and drop the Execute SQL Task to log error message



Procedure to create error record in the event table (if you do not have one please create this proc)

```
CREATE PROC SSIS.GetSSISEventLogData(@PackageId UNIQUEIDENTIFIER ,
                                     @Error NVARCHAR(MAX)  ,
                                     @Source NVARCHAR(100),
                                     @PackageName NVARCHAR(100)
                                     )
AS
BEGIN
            INSERT INTO [dbo].[SSIS_Event_Log]
            ([PackageID]
            ,[Error]
            ,[Source]
            ,[PackageName])
            VALUES
            (
             @PackageId  ,
         @Error  ,
         @Source ,
         @PackageName
         )

END
```

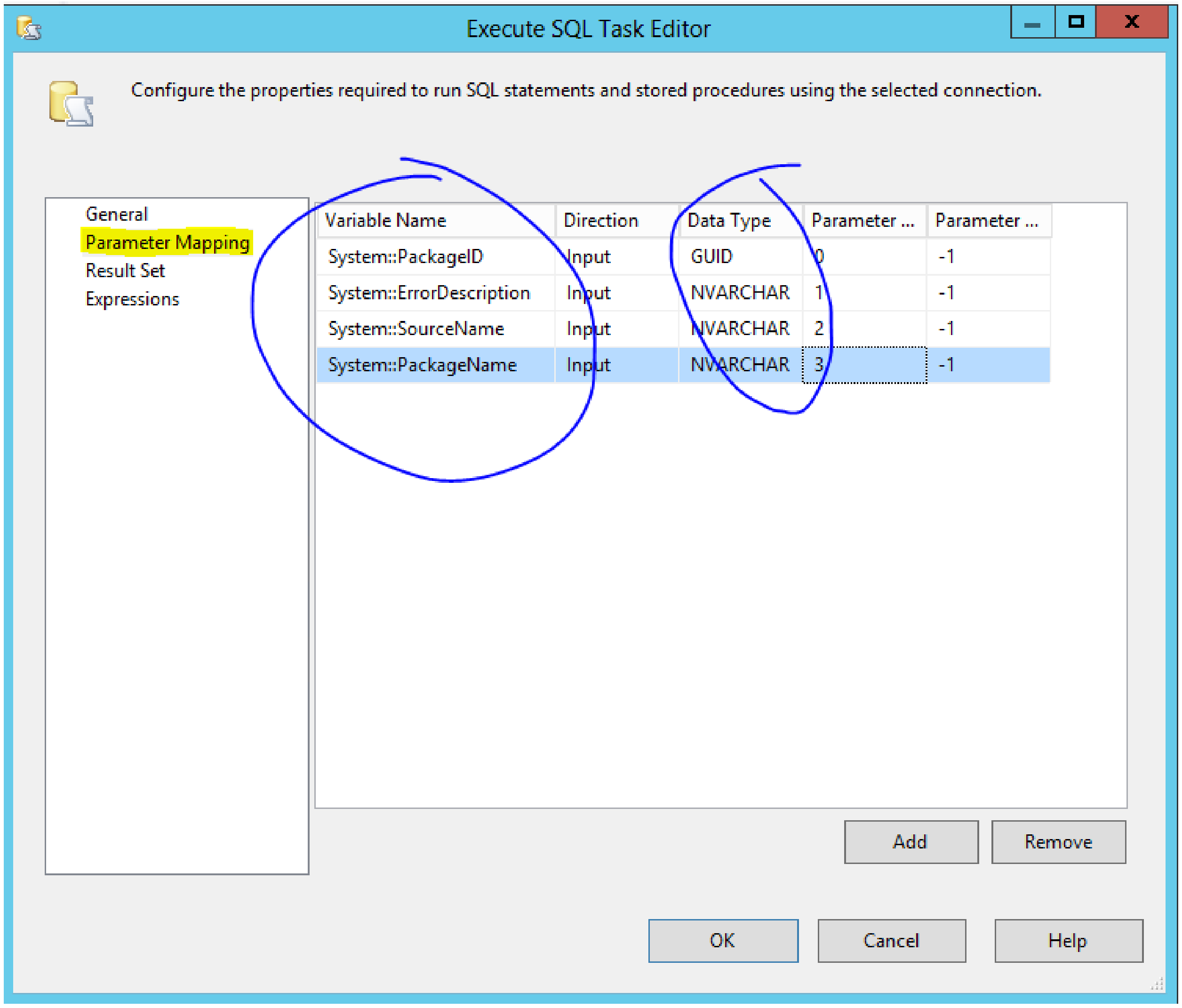
Configure the Execute SQL Task



General	
Name	**Execute SQL Task**
Description	**Execute SQL Task**
Options	
TimeOut	**0**
CodePage	**1252**
TypeConversionMode	**Allowed**
Result Set	
ResultSet	**None**
SQL Statement	
ConnectionType	**OLE DB**
Connection	**OLE DB Destination**
SQLSourceType	**Direct input**
SQLStatement	**EXEC SSIS.GetSSISEventLogData ?,?,?,?**
IsQueryStoredProcedure	False
BypassPrepare	**True**

Connection
Specifies the connection used to access the data.

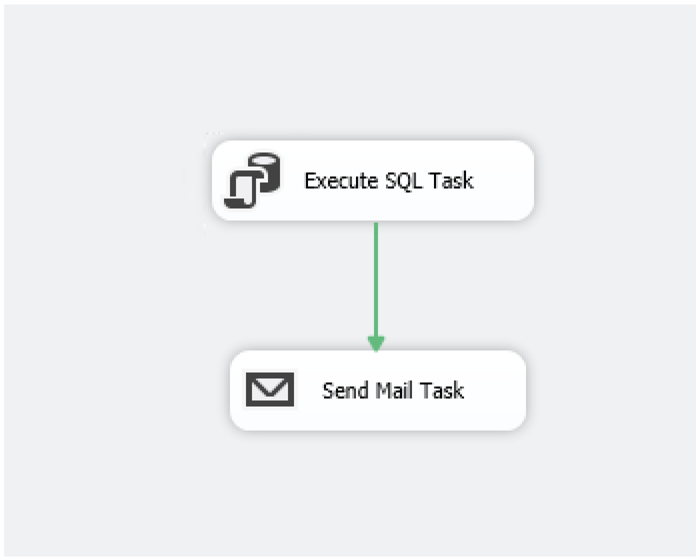
Configure the parameters



Configure the properties required to run SQL statements and stored procedures using the selected connection.

Variable Name	Direction	Data Type	Parameter ...	Parameter ...
System::PackageID	Input	GUID	0	-1
System::ErrorDescription	Input	NVARCHAR	1	-1
System::SourceName	Input	NVARCHAR	2	-1
System::PackageName	Input	NVARCHAR	3	-1

Then implement send mail task to send email notification to the user


Execute SQL Task
Send Mail Task