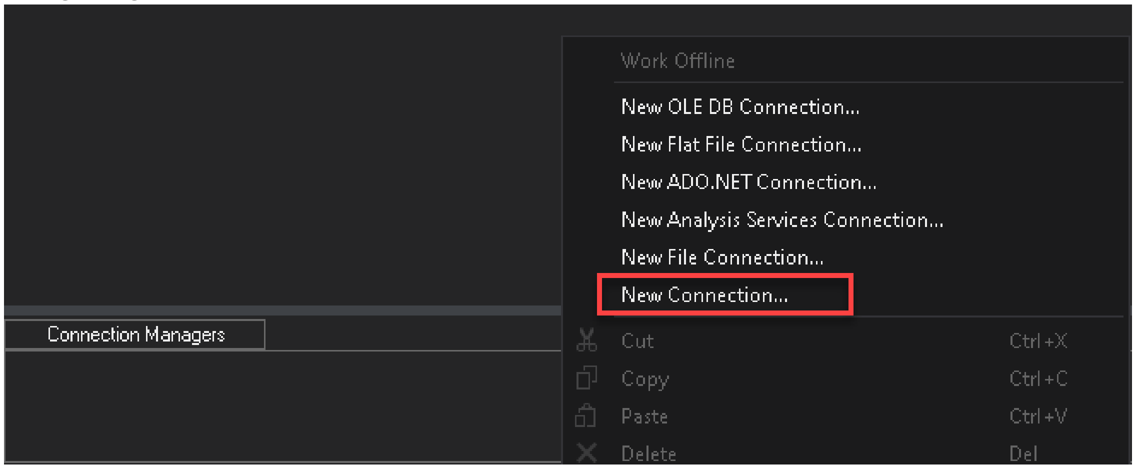
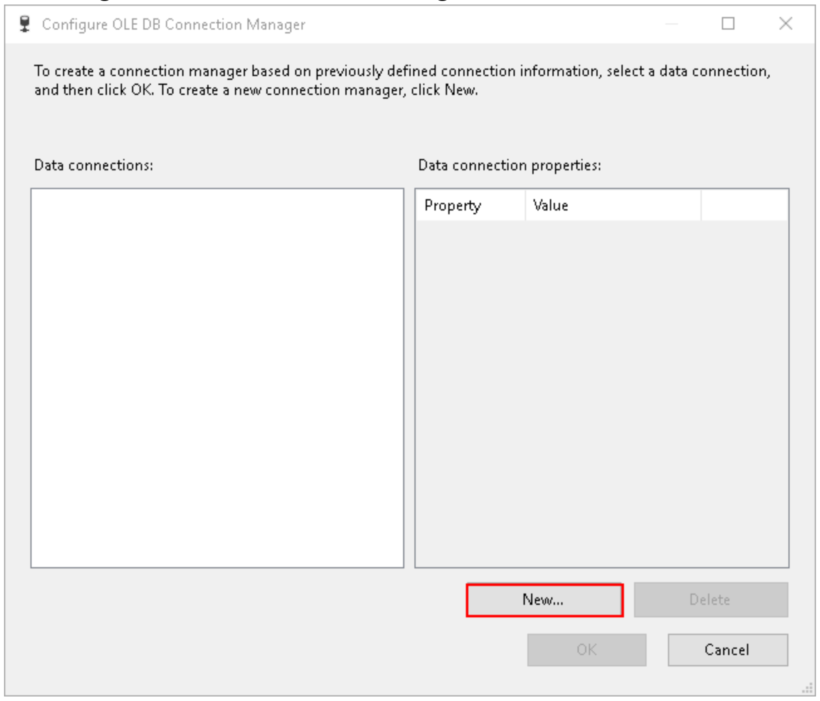
# SQL SERVER INTEGRATION SERVICES

## MODULE 04 – LAB 02: EVENT HANDLERS

1. Create new integration services project.
2. Set up a connection manager for the database. In the bottom center pane under Connection Manger, right-click select New Ole-DB Connection.

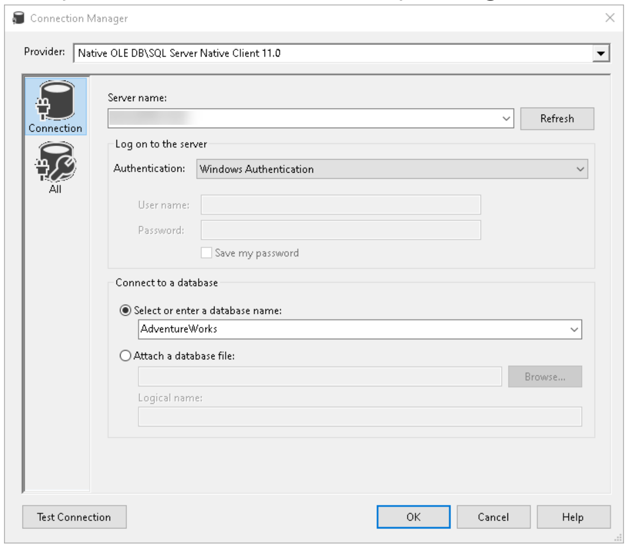


3. In Configure OLE DB Connect Manager, click New.


Configure OLE DB Connection Manager

To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

4. Setup connection to SQL Server pointing to the AdventureWorks database.



Connection Manager

Provider: Native OLE DB\SQL Server Native Client 11.0

Connection | All

Server name: [redacted] Refresh

Log on to the server

Authentication: Windows Authentication

User name:

Password:

Save my password

Connect to a database

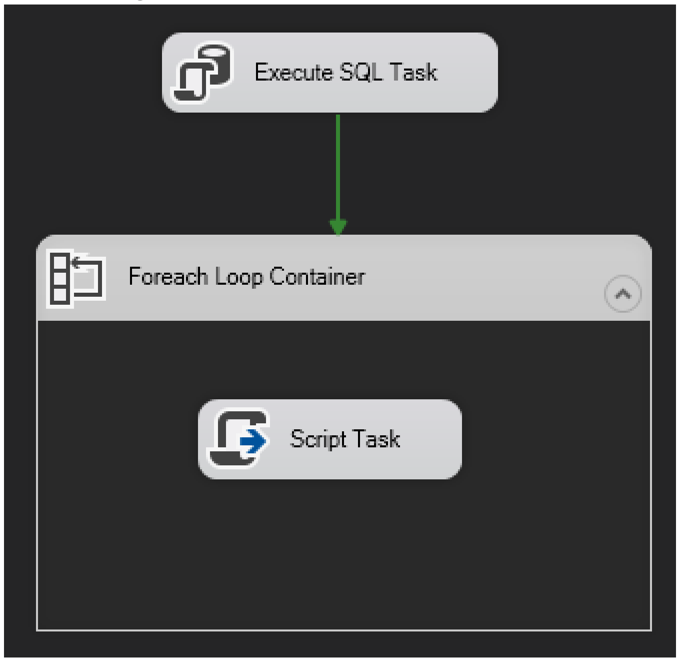
Select or enter a database name: AdventureWorks

Attach a database file: Browse...

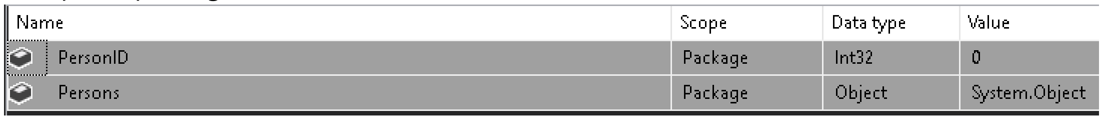
Logical name:

Test Connection | OK | Cancel | Help

5. Setup the tasks (Execute SQL Task, Foreach Loop Container, and Script Task). and link them as per the diagram below.



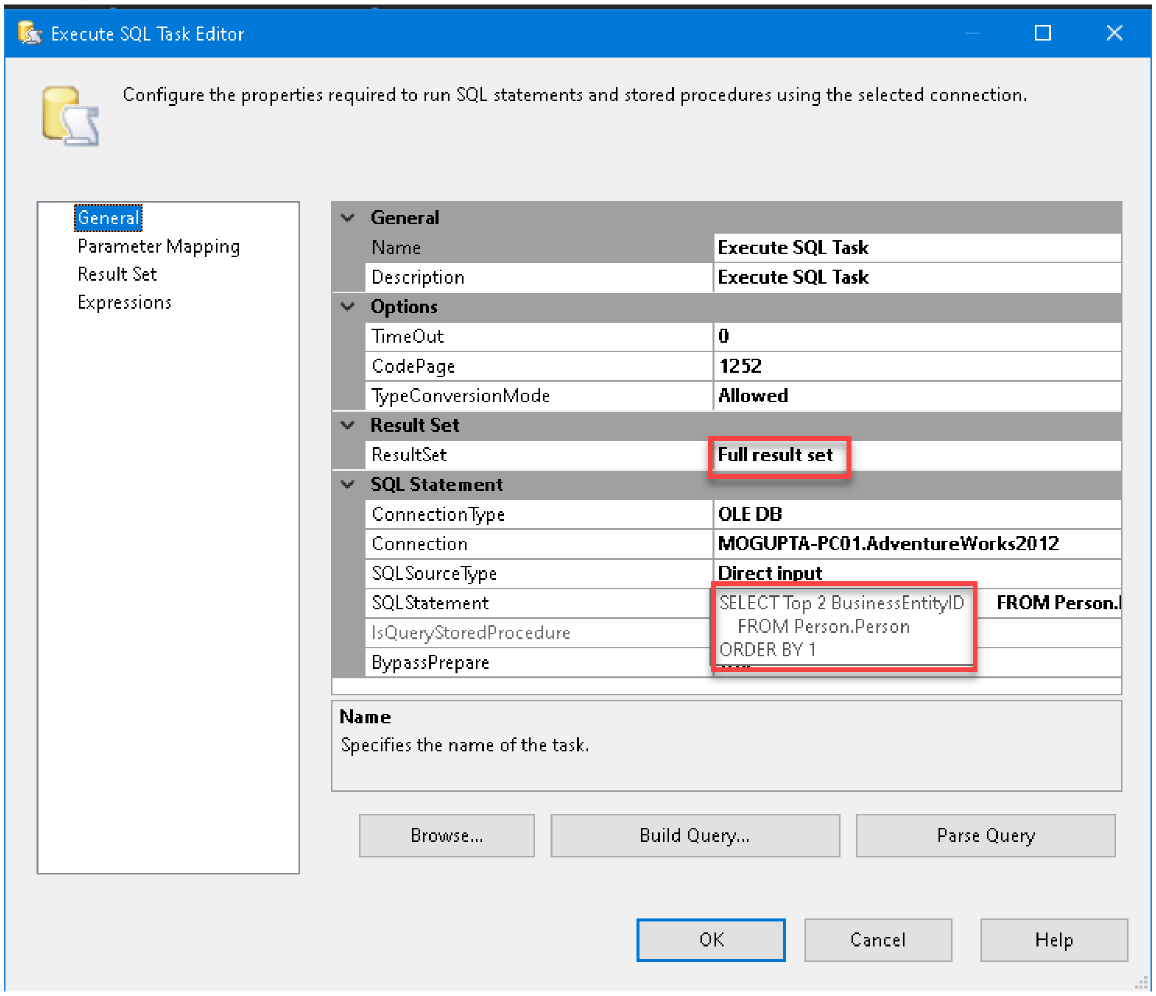
6. Setup two package variables (PersonID and Persons).



| Name | Scope | Data type | Value |
|---|---|---|---|
| PersonID | Package | Int32 | 0 |
| Persons | Package | Object | System.Object |

7. Configure the Execute SQL Task linking to connection created in Step #1.

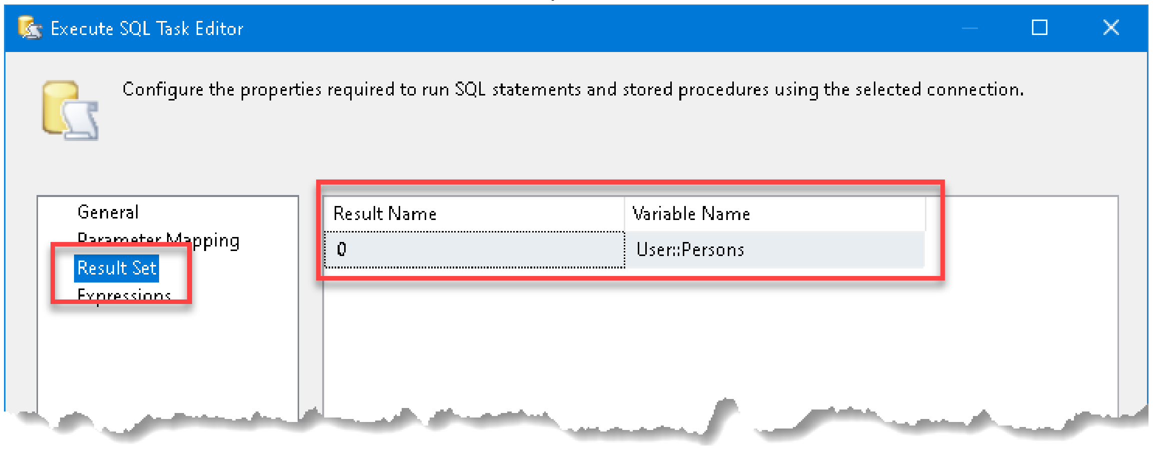
8. Execute the SQL Statement (SELECT TOP 2 BusinessEntityID FROM Person.Person ORDER BY 1), output the results to a "Full result set."



| | |
|---|---|
| **General** | |
| Name | Execute SQL Task |
| Description | Execute SQL Task |
| **Options** | |
| TimeOut | 0 |
| CodePage | 1252 |
| TypeConversionMode | Allowed |
| **Result Set** | |
| ResultSet | Full result set |
| **SQL Statement** | |
| ConnectionType | OLE DB |
| Connection | MOGUPTA-PC01.AdventureWorks2012 |
| SQLSourceType | Direct input |
| SQLStatement | SELECT Top 2 BusinessEntityID FROM Person.Person ORDER BY 1 |
| IsQueryStoredProcedure | |
| BypassPrepare | |

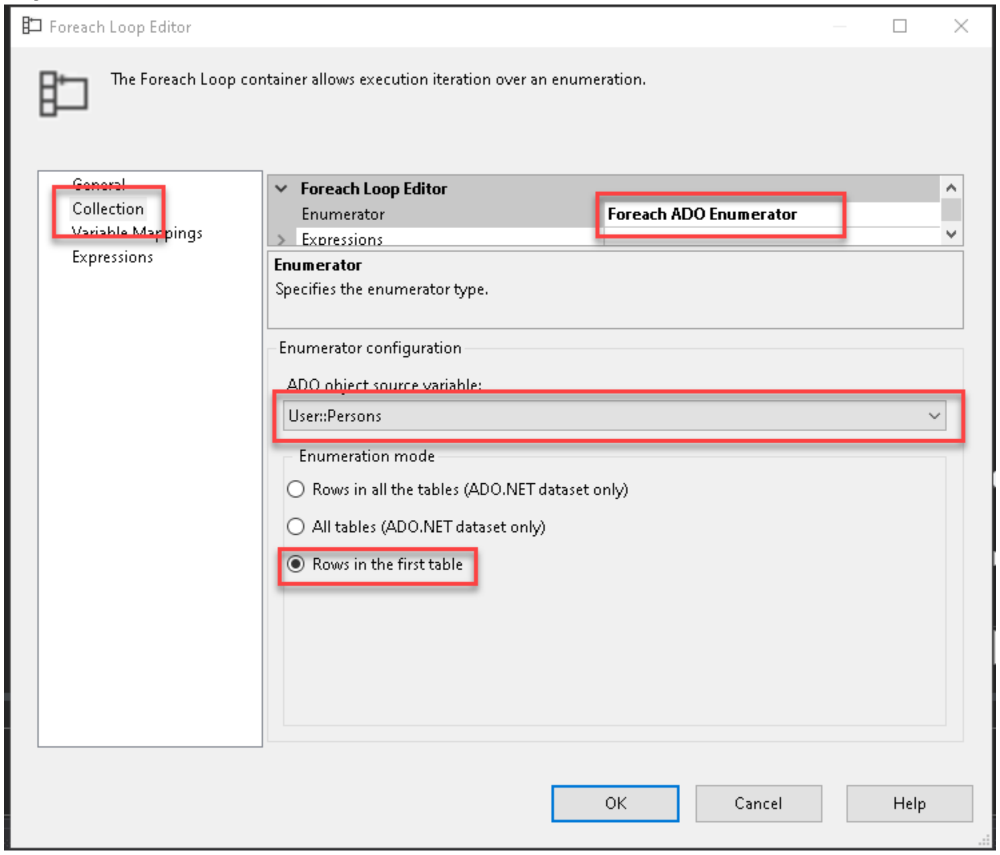
**Name**
Specifies the name of the task.

9. Save the result set to a variable created in step #6.


Execute SQL Task Editor
Configure the properties required to run SQL statements and stored procedures using the selected connection.
General
Parameter Mapping
Result Set
Expressions

| Result Name | Variable Name |
|---|---|
| 0 | User::Persons |

10. Configure the ForEach loop. Under Enumerator select "Foreach ADO Enumerator". From ADO object select the variable User::Persons.



Enumerator: Foreach ADO Enumerator

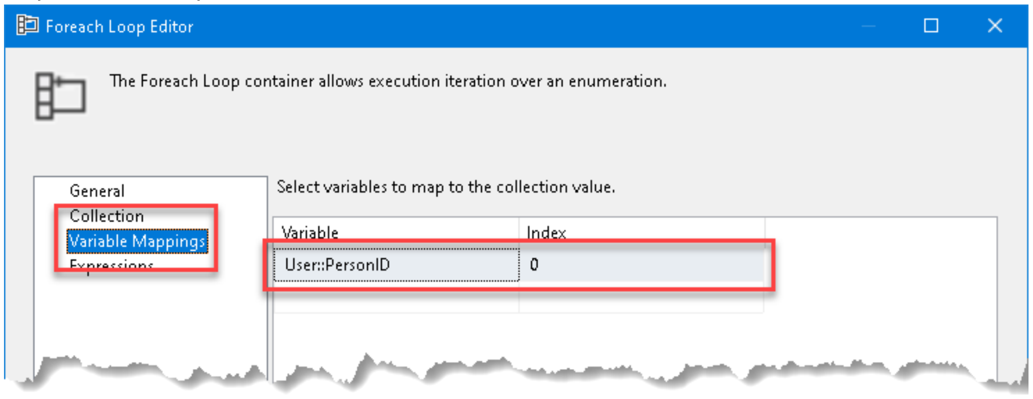
Enumerator
Specifies the enumerator type.

Enumerator configuration

ADO object source variable: User::Persons

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

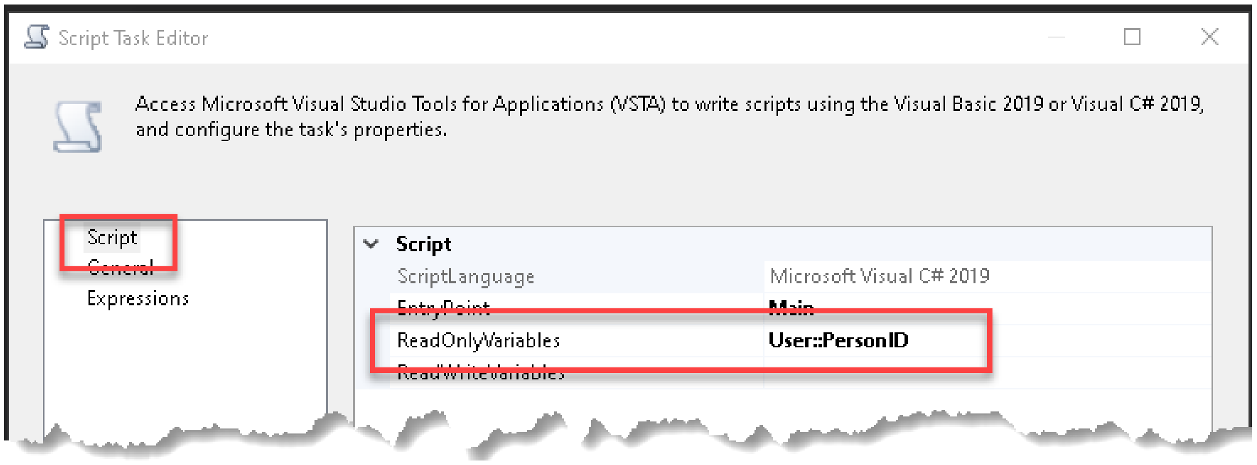
11. Capture the output of each row in result set and save it to variable User::PersonID.


Foreach Loop Editor
The Foreach Loop container allows execution iteration over an enumeration.
General
Collection
Variable Mappings
Expressions

Select variables to map to the collection value.

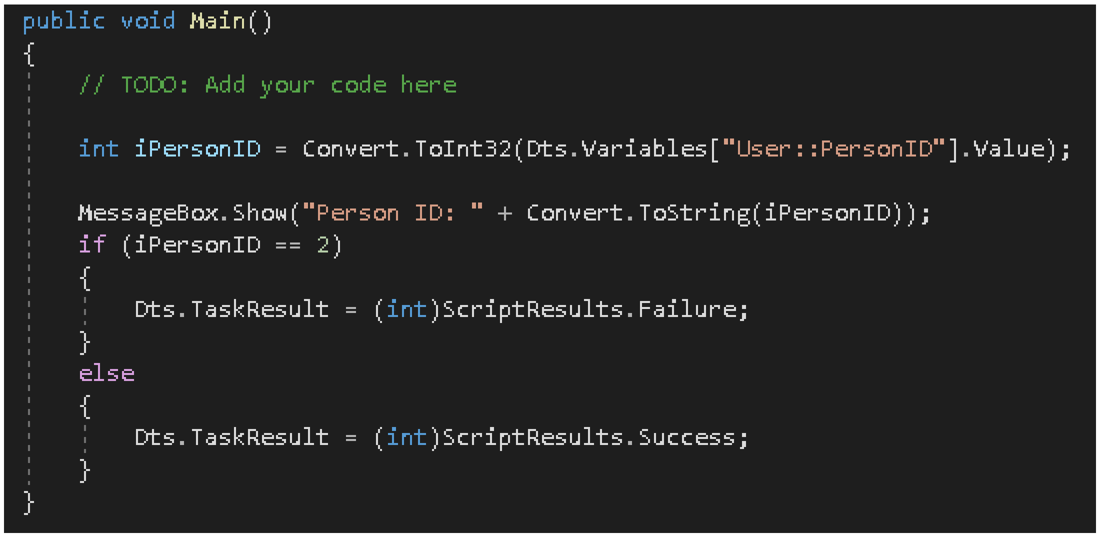
| Variable | Index |
|---|---|
| User::PersonID | 0 |

12. Modify the script task, pass in User::PersonID as ReadOnlyVariable. Add the following code in the script:



Access Microsoft Visual Studio Tools for Applications (VSTA) to write scripts using the Visual Basic 2019 or Visual C# 2019, and configure the task's properties.

```
ScriptLanguage        Microsoft Visual C# 2019
EntryPoint            Main
ReadOnlyVariables     User::PersonID
ReadWriteVariables
```

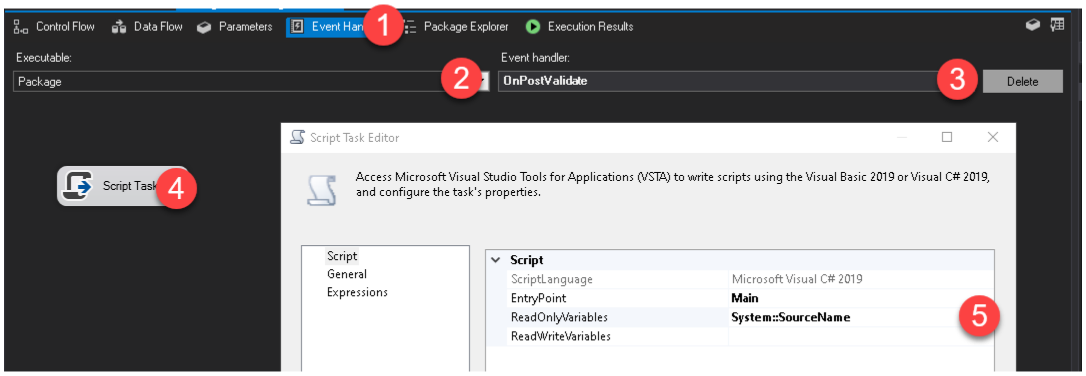


```
public void Main()
{
    // TODO: Add your code here

    int iPersonID = Convert.ToInt32(Dts.Variables["User::PersonID"].Value);

    MessageBox.Show("Person ID: " + Convert.ToString(iPersonID));
    if (iPersonID == 2)
    {
        Dts.TaskResult = (int)ScriptResults.Failure;
    }
    else
    {
        Dts.TaskResult = (int)ScriptResults.Success;
    }
}
```

```
int iPersonID = Convert.ToInt32(Dts.Variables["User::PersonID"].Value);

MessageBox.Show("Person ID: " + Convert.ToString(iPersonID));
if (iPersonID == 2)
{
     Dts.TaskResult = (int)ScriptResults.Failure;
}
else
{
     Dts.TaskResult = (int)ScriptResults.Success;
}
```

13. After setting up the script test, it out. It should come back with two messages and fail because of result status returned from script.

14. Next add the following steps for each of the events under package.



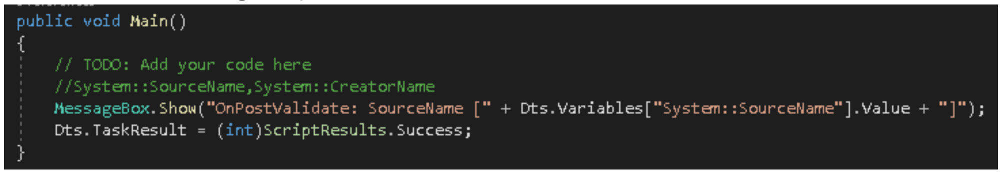
Executable: Package

Event handler: OnPostValidate

Access Microsoft Visual Studio Tools for Applications (VSTA) to write scripts using the Visual Basic 2019 or Visual C# 2019, and configure the task's properties.

| Script | |
|---|---|
| ScriptLanguage | Microsoft Visual C# 2019 |
| EntryPoint | Main |
| ReadOnlyVariables | System::SourceName |
| ReadWriteVariables | |

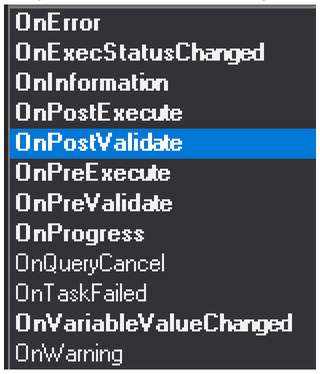
a. Go to Event Handler under package.
b. Select Package from Executable.
c. Select first event in list.
d. Create Script Task.
e. In script task pass in ReadOnlyVariable, System::SourceName.
f. Create the following script.



```
public void Main()
{
    // TODO: Add your code here
    //System::SourceName,System::CreatorName
    MessageBox.Show("OnPostValidate: SourceName [" + Dts.Variables["System::SourceName"].Value + "]");
    Dts.TaskResult = (int)ScriptResults.Success;
}
```

```
MessageBox.Show("OnPostValidate: SourceName [" +
Dts.Variables["System::SourceName"].Value + "]");
```

g. Repeat the above steps for each event under Package.



OnError
OnExecStatusChanged
OnInformation
OnPostExecute
OnPostValidate
OnPreExecute
OnPreValidate
OnProgress
OnQueryCancel
OnTaskFailed
OnVariableValueChanged
OnWarning

15. Execute the package and review what order the events are raised. Notice how events are bubbled up from deepest execution “Script Task” in Container to Package level.