Business Intelligence Development Studio (SSIS) and U2 Toolkit for .NET v.2.1.0

# Overview

Business Intelligence Development Studio is the environment that you will use to develop packages for data extraction, transformation, and loading (ETL) in Integration Services (SSIS). SSIS includes graphical tools and wizards for building and debugging packages; tasks for performing workflow functions such as FTP operations, executing SQL statements, and sending e-mail messages; data sources and destinations for extracting and loading data; transformations for cleaning, aggregating, merging, and copying data; a management service, the Integration Services service for administering package execution and storage; and application programming interfaces (APIs) for programming the Integration Services object model.

In this tutorial, you will learn how to use SSIS Designer to create a simple Microsoft SQL Server Integration Services package. The package that you create takes data from a U2 Database, reformats the data, and then inserts the reformatted data into a SQL Server table

U2 Toolkit for .NET v2.1.0 supports:

* SQL Access
* Native Access

U2 Toolkit for .NET v2.1.0 is Connection Manager in SSIS Project.

# Requirements

|  |  |
| --- | --- |
| **U2** | **Version** |
| UniVerse | **10.3 or later** |
| UniData | **7.1 or later** |

|  |  |
| --- | --- |
| **Microsoft** | **Version** |
| Visual Studio | * **2010 SP1** * **2012 Update 4** * **2013 Update 2 RC** |
| Business Intelligence Development Studio (BIDS)  (SSIS and SSRS) | * **BIDS 2010** * **BIDS 2012** |

|  |
| --- |
| **Configure ProviderDescriptors.xml file** |
| 1. In VS2012, open C:\Program Files\Microsoft SQL Server\110\DTS\ProviderDescriptors\ ProviderDescriptors.xml file. 2. Add U2Connection entry. For example (See Appendix A) |

# Creating the Project and Basic Package

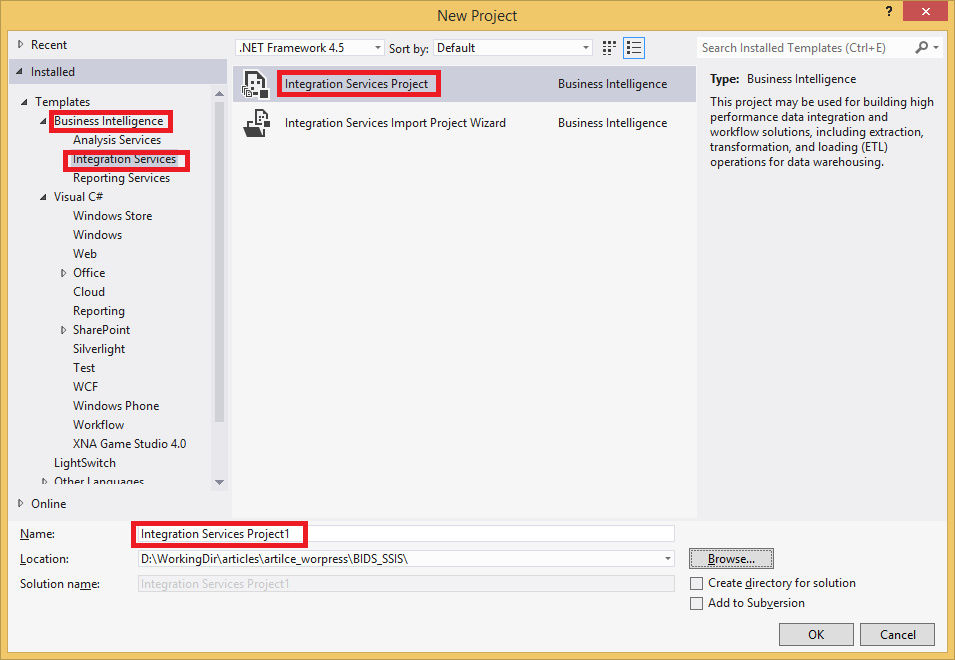
This sample code will show how to create a simple ETL package that extracts data from a U2 Database source and writes that data to the SQL Server Table. It will show how to create new packages, add and configure data source and destination connections, and work with new control flow and data flow components.

## Step 1: Creating a New Integration Services Project

The first step in creating a package in Integration Services is to create an Integration Services project. This project includes the templates for the objects — data sources, data source views, and packages — that you use in a data transformation solution.

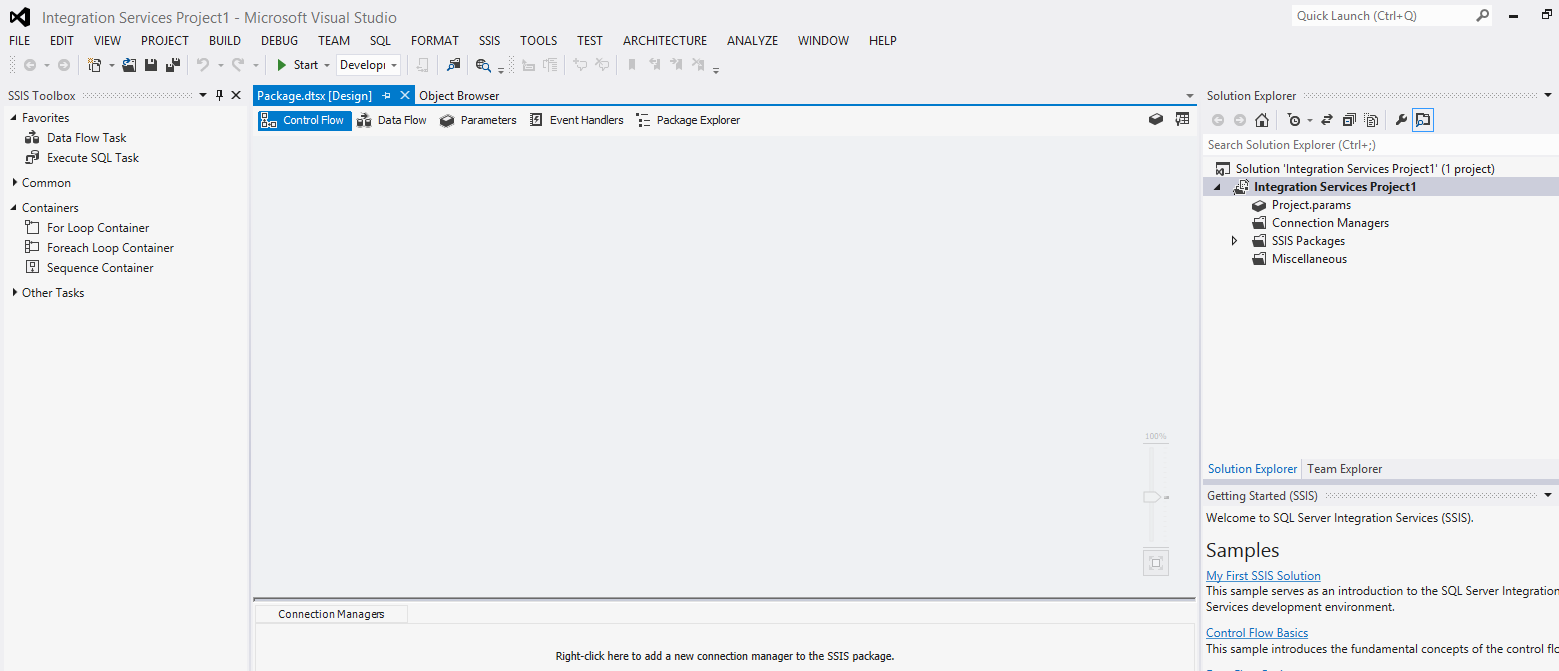
To create a new Integration Services project

1. On the Start menu, point to All Programs, point to Microsoft SQL Server, and click SQL Server SQL Server Data Tools.
2. On the File menu, point to New, and click Project to create a new Integration Services project.
3. In the New Project dialog box, select Integration Services Project in the Templates pane.
4. In the Name box, change the default name to SSIS Tutorial. Optionally, clear the Create directory for solution check box.
5. Accept the default location, or click Browse to browse to locate the folder you want to use. In the Project Location dialog box, click the folder and click Select Folder.
6. Click OK.
7. By default, an empty package, titled Package.dtsx, will be created and added to your project.



The new project screen contains the following:

* Tool Box on left side bar
* Solution Explorer on upper right bar
* Property Window on lower right bar
* Control flow, data flow, event Handlers, Package Explorer in tab windows
* Connection Manager Window in the bottom



## Step 2: Adding and Configuring a ADO.NET Connection Manager (U2 Source)

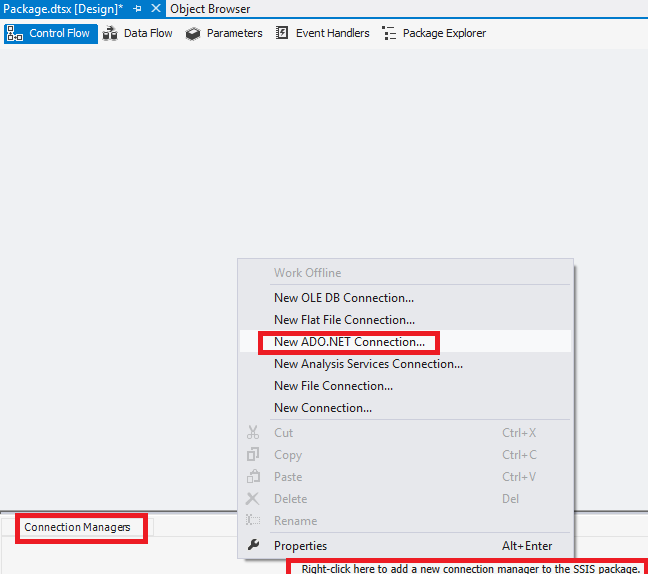
In this task, you add an ADO.NET connection manager to the package that you just created. ADO.NET connection manager enables a package to extract data from any ADO.NET source such as U2 Database. Using the ADO.NET connection manager, you can specify User, Password, Server, Database, ServerType and AccessMode.

If AccessMode is Uci, it uses U2 Toolkit for .NET SQL Access.

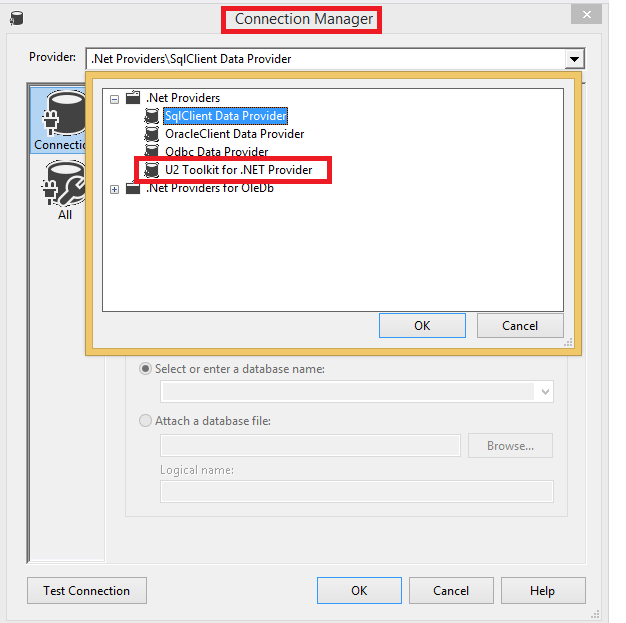
If AccessMode is Native, it uses U2 Toolkit for .NET Native Access.

**To add a ADO.NET connection manager to the SSIS package**

Right-click anywhere in the Connection Managers area, and then click New ADO.NET Connection.

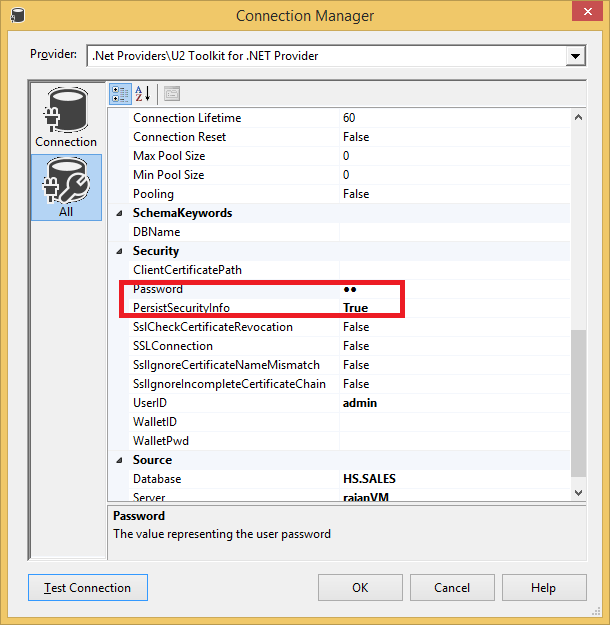


In Connection Manager, select U2 Toolkit for .NET Provider

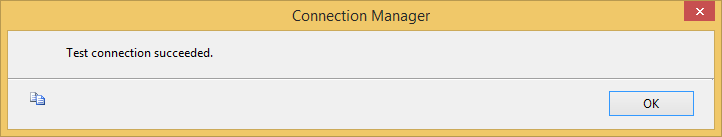


Enter connection string parameters.

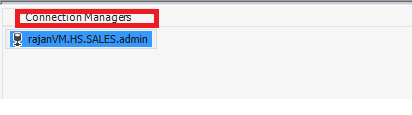
Requires: User, Password, Server, Database, ServerType, AccessMode, PersistSecurityInfo



Test Connection



You will see ADO.NET Connection Manager (U2 Database) in Connection Managers Pane.



## Step 3: Adding and Configuring a ADO.NET Connection Manager (SQL Server Destination)

After you have added an ADO.NET connection manager to connect to the U2 data source, the next task is to add an ADO.NET connection manager to connect to the SQL Server destination.

In this lesson, you will create an ADO.NET connection manager that uses Windows Authentication to connect to the local instance of SQL Server.

To add and configure an ADO.NET Connection Manager

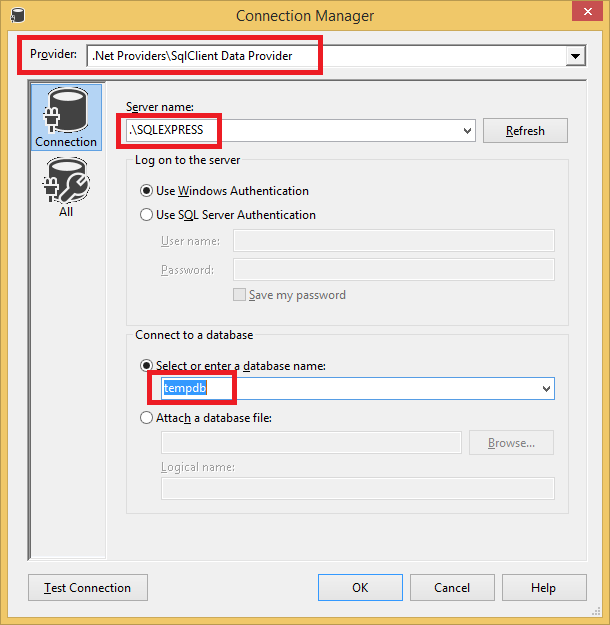
Right-click anywhere in the Connection Managers area, and then click New ADO.NET Connection.

In the Configure ADO.NET Connection Manager dialog box, click New.

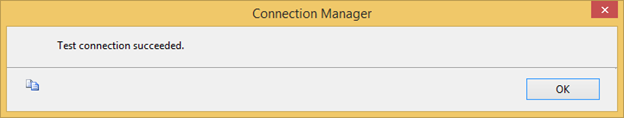
Select “SqlClient Data Provider” from Provider combo box.

For Server name, enter .\SQLEXPRESS.

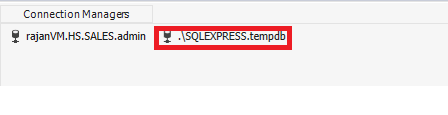
For Database name, select ‘tempdb’.



Test the connection



You will see ADO.NET Connection Manager (Destination) in Connection Managers Pane.

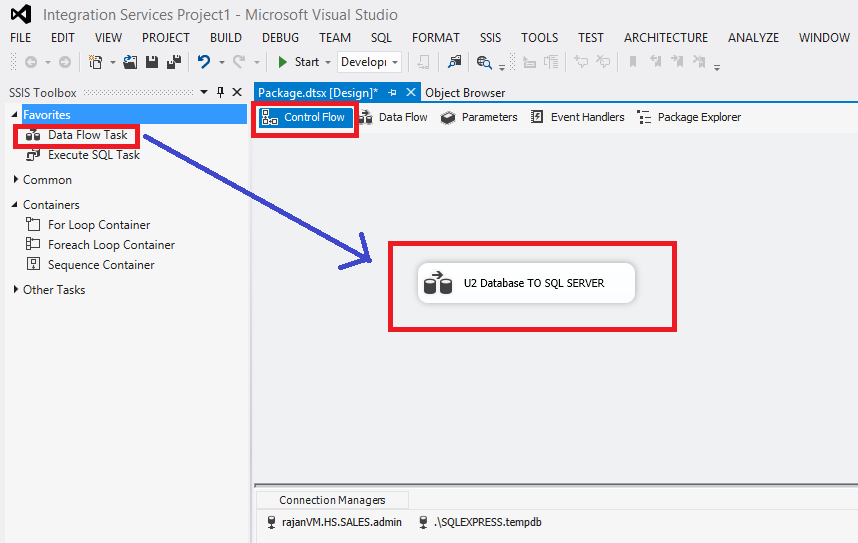


## Step 4: Adding a Data Flow Task to the Package

After you have created the connection managers for the source and destination data, the next task is to add a Data Flow task to your package. The Data Flow task encapsulates the data flow engine that moves data between sources and destinations, and provides the functionality for transforming, cleaning, and modifying data as it is moved. The Data Flow task is where most of the work of an extract, transform, and load (ETL) process occurs.

**To add a Data Flow task**

1. Click the Control Flow tab.
2. In the Toolbox, expand Control Flow Items, and drag a Data Flow Task onto the design surface of the Control Flow tab.
3. On the Control Flow design surface, right-click the newly added Data Flow Task, click Rename, and change the name to “**U2 Database TO SQL Server**”.



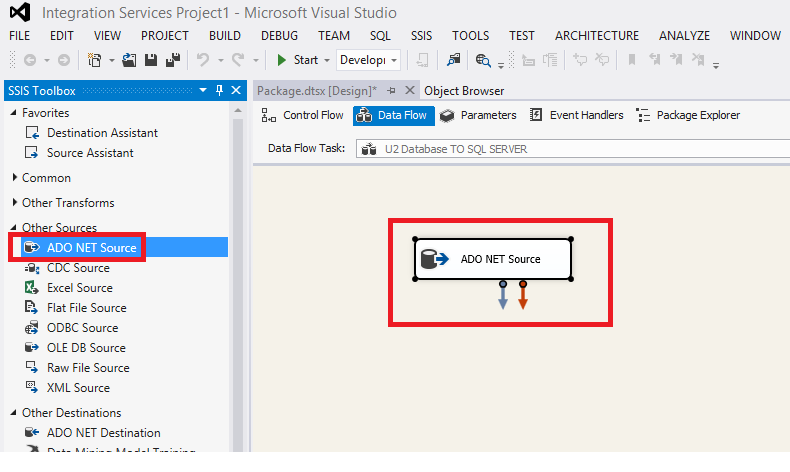
## Step 5: Adding and Configuring the ADO.NET Source

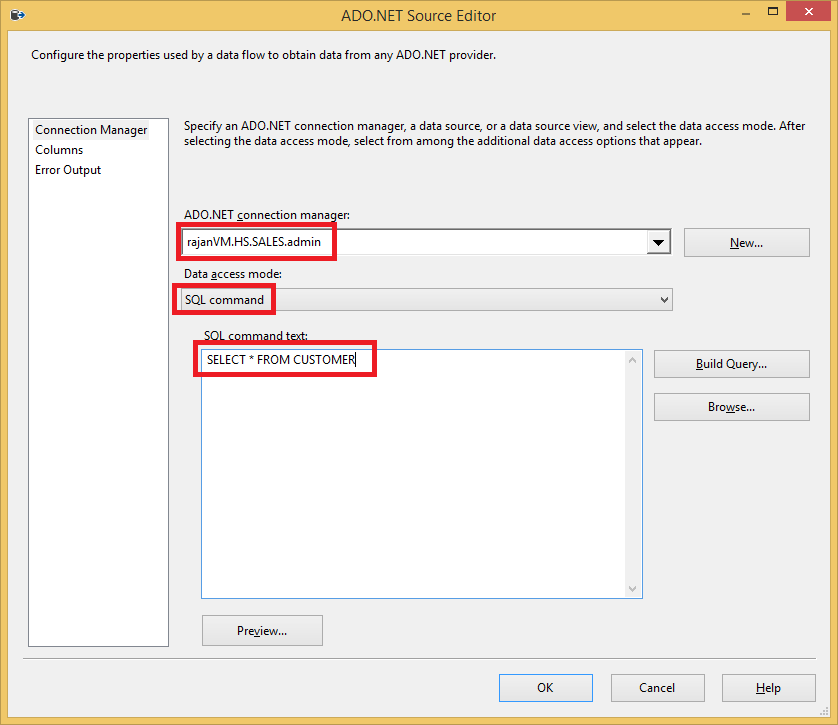
In this task, you will add and configure an ADO.NET source to your package. The ADO NET source consumes data from a .NET provider and makes the data available to the data flow.

For this tutorial, you will configure the ADO.NET source to use the U2 Database ADO.NET Source Data connection manager that you previously created.

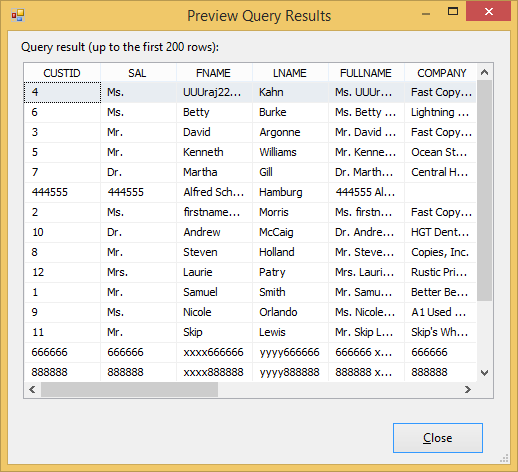
**To add a ADO.NET Source component**

1. Open the Data Flow designer, either by double-clicking the “**U2 Database TO SQL Server**” data flow task or by clicking the Data Flow tab.
2. In the Toolbox, expand Data Flow Sources, and then drag an ADO.NET Source onto the design surface of the Data Flow tab.
3. On the Data Flow design surface, right-click the newly added ADO.NET Source, click Rename, and change the name to “**Extract Customer Table**”.
4. Double-click the ADO.NET source to open the ADO.NET Source Editor Dialog box.
5. Select Data Access Mode: SQL Command
6. Enter for SQL Command Text: **SELECT \* FROM CUSTOMER**





1. Preview data

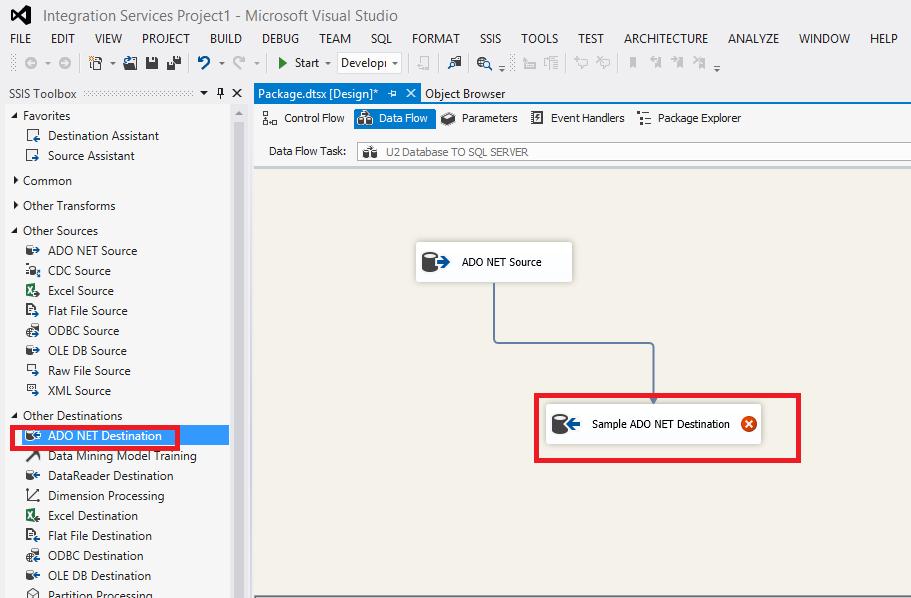


## Step 6: Adding and Configuring the ADO.NET Destination

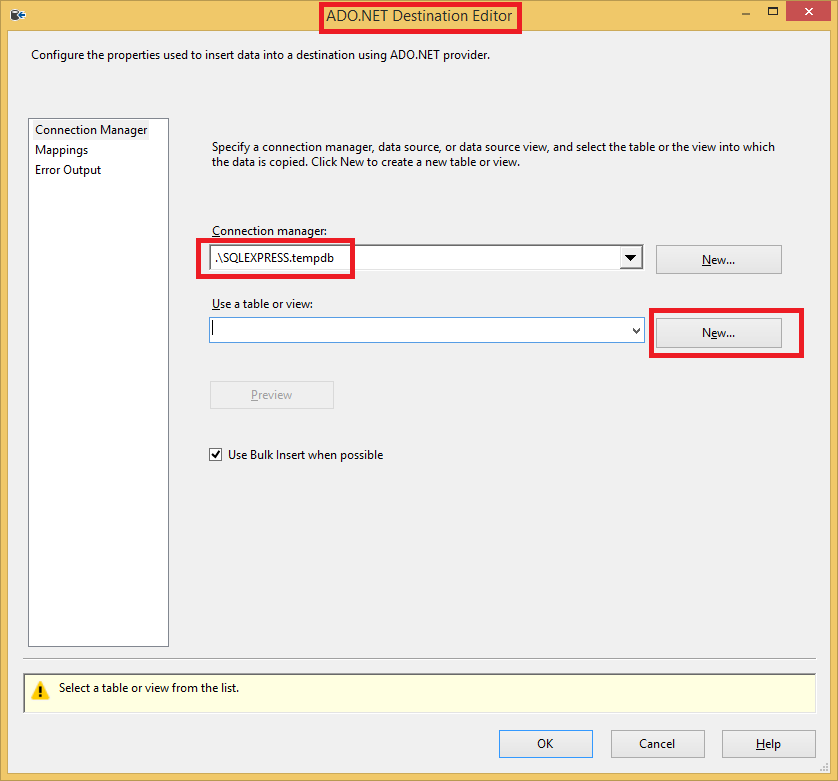
Your package now can extract data from the ADO.NET source. The next task is to actually load the transformed data into the destination. To load the data, you must add an ADO.NET destination to the data flow.

To add and configure the sample ADO.NET destination

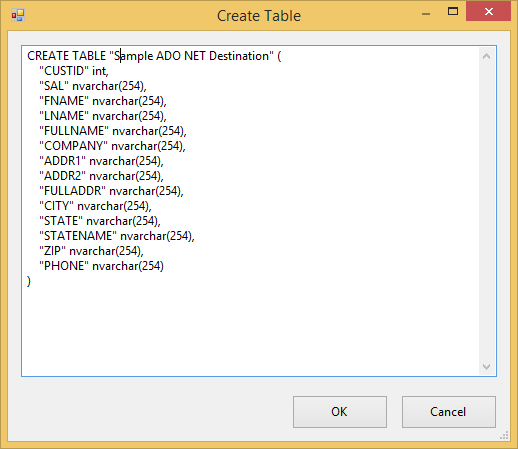
1. In the SSIS Toolbox, expand Other Destinations, and drag ADO.NET Destination onto the design surface of the Data Flow tab. Place the ADO.NET destination directly below the “**Extract Customer Table**”.
2. Click “**Extract Customer Table”** and drag the green arrow over to the newly added ADO.NET Destination to connect the two components together.
3. On the Data Flow design surface, click ADO.NET Destination in the newly added ADO.NET Destination component, and change the name to “**Sample ADO NET Destination**”.



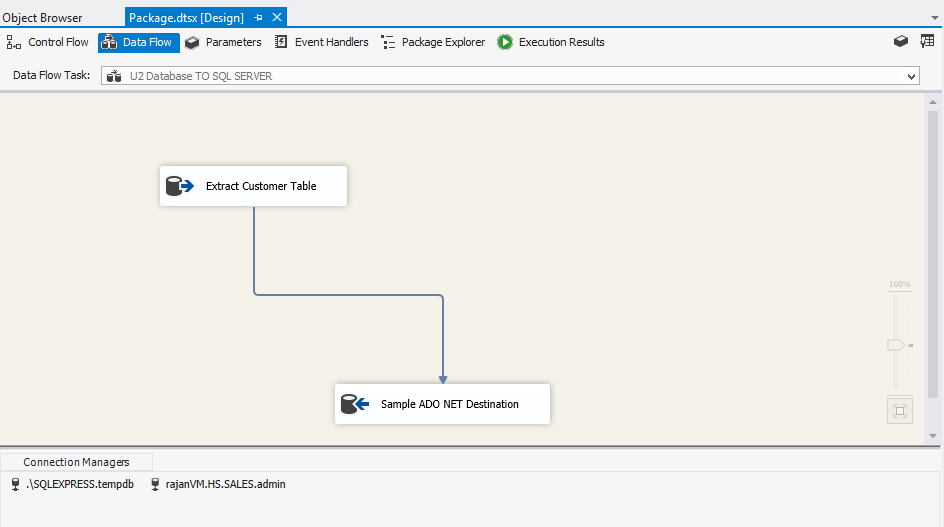
1. Double-click Sample ADO.NET Destination.
2. In the ADO.NET Destination Editor dialog box, ensure that SQLEXPRESS is selected in the ADO.NET Connection manager box.



1. Click “New…”
2. Take everything default



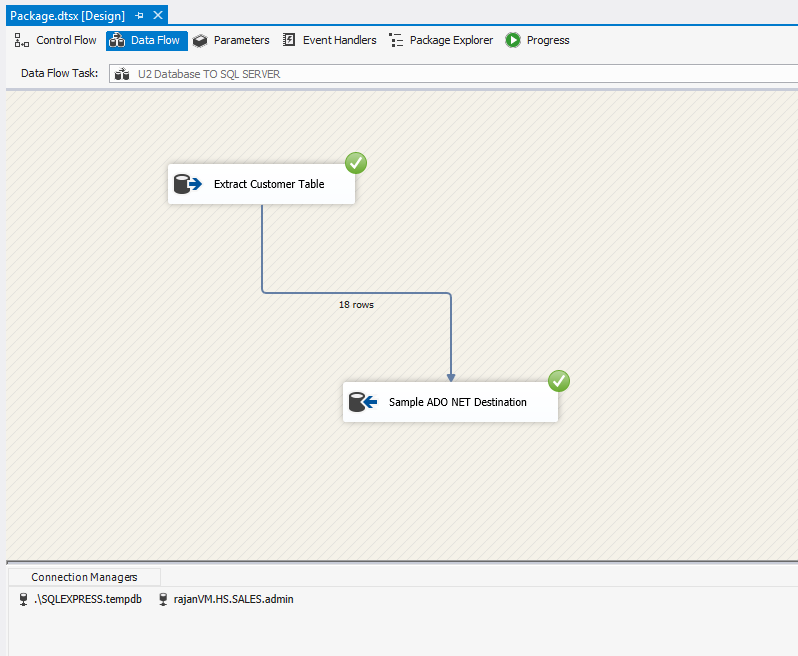
1. Click Preview. You will see empty table. Close it.
2. Click Mapping. Verify all columns are mapped correctly.
3. Close ADO.NET Destination Editor



## Step 7: Testing the Package

**To run the package**

1. On the Debug menu, click Start Debugging. The package will run, resulting in some rows successfully added into the “**Sample ADO NET Destination”** fact table in “tempdb”.
2. After the package has completed running, on the Debug menu, click Stop Debugging.



## Resources

[SSIS Tutorial: Creating a Simple ETL Package](http://msdn.microsoft.com/en-us/library/ms169917.aspx)

# Source Code

* [See Github Rocket U2 Server Lab](https://github.com/RocketSoftware/u2-servers-lab/tree/master/U2-Toolkit)

## Appendix A

|  |
| --- |
| <dtm:ProviderDescriptor SourceType="U2.Data.Client.U2Connection">  <dtm:SchemaNames  TablesSchemaName="Tables"  ColumnsSchemaName="Columns"  ViewsSchemaName="Views"  />  <dtm:TableSchemaAttributes  TableCatalogColumnName="TABLE\_CAT"  TableSchemaColumnName="TABLE\_SCHEM"  TableNameColumnName="TABLE\_NAME"  TableTypeColumnName="TABLE\_TYPE"  TableDescriptor="TABLE"  ViewDescriptor="VIEW"  SynonymDescriptor ="SYNONYM"  NumberOfTableRestrictions="3"  />  <dtm:ColumnSchemaAttributes  NameColumnName = "COLUMN\_NAME"  OrdinalPositionColumnName="ORDINAL\_POSITION"  DataTypeColumnName = "DATA\_TYPE"  MaximumLengthColumnName = "COLUMN\_SIZE"  NumericPrecisionColumnName = "PRECISION"  NumericScaleColumnName = "SCALE"  NullableColumnName="IS\_NULLABLE"  NumberOfColumnRestrictions="4"  />  <dtm:Literals  PrefixQualifier="&quot;"  SuffixQualifier="&quot;"  CatalogSeparator="."  SchemaSeparator="."  />    </dtm:ProviderDescriptor> |