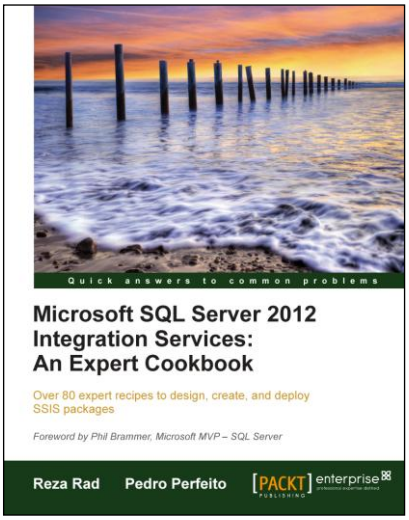
[PACKT] PUBLISHING enterprise professional expertise distilled

Microsoft SQL Server 2012 Integration Services: An Expert Cookbook

Reza Rad

Pedro Perfeito



Chapter No. 1

"Getting Started with SQL Server Integration Services"

In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.1 "Getting Started with SQL Server Integration Services"

A synopsis of the book's content

Information on where to buy this book

About the Authors

Reza Rad is an author, trainer, speaker, and consultant. He has a BSc in Computer Engineering; he has more than 10 years' experience in programming and development mostly on Microsoft technologies. He received the Microsoft Most Valuable Professional (MVP) award in SQL Server in 2011 and 2012 for his dedication in Microsoft BI and specially SSIS. He has been working on the Microsoft BI suite for more than six years. He is an SSIS/MSBI/.NET Trainer and also software and BI Consultant at some companies and institutes. His articles on different aspects of technologies, specially on SSIS, can be found on his blog `http://www.rad.pasfu.com`.

He was the co-author of *SQL Server MVP Deep Dives Volume 2*. He is one of the active members on online technical forums such as MSDN and Experts-Exchange. He is a Microsoft Certified Professional (MCP); Microsoft Certified Technology Specialist (MCTS) and Microsoft Certified IT Professional (MCITP) in Business Intelligence (BI). His e-mail address is `a.raad.g@gmail.com`.

I would like to thank my wife who has been a wonderful supporter in writing this book; she encouraged me a lot to complete this book, she was a light during my difficult moments.

I would also like to thank my parents and sister, who were my teachers for many years of my life.

I would like to thank Pedro, my good friend who helped a lot in writing this book. He did a good job in completing this book in his busy hours with full-time job and teaching.

Pedro Perfeito was born in 1977 in Portugal and currently works as a BI Senior Consultant and Developer at Novabase. He's also an invited teacher in master and short-master BI degrees at IUL-ISCTE (Lisbon) and at Universidade Portucalense (UPT-Porto) respectively. He received the Microsoft award Microsoft Most Valuable Professional (MVP) in 2010, 2011, and 2012 for all his dedication and contribution in helping theoretical and practical issues in the various BI communities. He is also the co-author of *SQL Server MVP Deep Dives Volume 2*. He has several Microsoft certifications including MCP, MCSD, MCTS-Web, MCTS-BI, and MCITP-BI. He also has worldwide certifications in the area of BI provided by TDWI/CBIP (The Data Warehouse Institute, `http://www.tdwi.org`). He's currently preparing for his PhD degree on BI. For further details you can visit his personal blog at `http://www.pedrocgd.blogspot.com` or even contact him directly at `pperfeito@hotmail.com`.

I would like to express my gratitude to all teams at Packt who trusted me—a Portuguese author—and helped me complete this book. I would like to thank my friend and co-author of this book Reza Rad because without him this book would not have been possible.

I have furthermore to thank Barbara Chambel for all the support she gave me since the first moment at Novabase, to Luis Ferreira (Project Manager at Banco de Portugal) and Simão Fernandes (ex-student and colleague at Novabase) for all hints and complaints from the previous SSIS version (you both know which ones I mean!) and for all my Master BI students from Universidade Portucalense (Oporto) and from ISCTE-IUL (Lisbon) who have directly and indirectly motivated me in this challenge.

I am deeply indebted to Dr. Maria José Trigueiros for all the encouragement to go inside this amazing world of Business Intelligence and make my dream come true. She's not physically with us but she will be remembered for ever.

Especially, I would like to give my special thanks to my family and my girlfriend Joana whose patient love helped me to complete this work!

Thanks to all who I haven't mentioned here and who believed in me, even more than myself.

Microsoft SQL Server 2012 Integration Services: An Expert Cookbook

Microsoft SQL Server 2012 Integration Services: An Expert Cookbook is a complete guide for everyone, from a novice to a professional in Integration Services 2012. SQL Server Integration Services is an ETL tool, which stands for Extract Transform and Load. There is a need for a data transfer system in all operational systems these days, and SSIS is one of the best data transfer tools. In this book, all aspects of SSIS 2012 are discussed with lots of real-world scenarios to help readers to understand usage of SSIS in every environment.

What This Book Covers

Chapter 1, Getting Started with SQL Server Integration Services, provides an overview of the ETL concepts and ETL terminologies, why ETL is needed in the technology world, and what problems ETL will solve. Then an overview of SSIS as an ETL tool is provided to help readers to get an overall view of the other parts of the book.

Chapter 2, Control Flow Tasks, explores all Control Flow Tasks with real-world samples of each Task. The reader will learn what each Task stands for, what is its usage, real-world scenarios, and the new tasks available in SSIS 2012.

Chapter 3, Data Flow Task Part 1—Extract and Load, explains the data sources and data destinations under the Data Flow Task. Data Flow Task is the most functional part of SSIS, to which an SSIS Developer probably dedicates most time.

Chapter 4, Data Flow Task Part 2—Transformations, explores the transformations used to apply data quality and business rules that are essential to prepare data loaded into destinations. Data Flow Task provides an easy way to transform source data into the form needed by its destination in several different ways.

Chapter 5, Data Flow Task Part 3—Advanced Transformation, briefly discusses Advanced Transformations. In real-world scenarios, different data sources don't provide the same structure, so there is a need to unify them in a unique structure. There are some transformations in SSIS Data Flow Task that use complex ways to apply such changes on data stream. We call them Advanced Transformations.

Chapter 6, Variables, Expressions, and Dynamism in SSIS, describes how SSIS works with dynamism with the aid of expressions, what are the limitations of some tasks in dynamism, and what are the alternative solutions. SSIS as an executable unit needs to have a structure for declaring in-memory variables and store some data in memory to pass between Tasks through the execution phase. Besides the variables, there is a built-in statement language in SSIS components and Tasks to do many operations such as data conversion, data splitting based on a condition, or creating text filenames based on date. In this chapter, readers will learn how to work with variables and expressions in many scenarios. Dynamism is the most powerful aspect of an ETL tool in data transfer operations.

Chapter 7, Containers and Precedence Constraints, explains three types of containers and precedence constraint in the SSIS Control Flow, which help developers to control the flow of task execution. All of these containers and the precedence constraints are covered in this chapter with real-world samples.

Chapter 8, Scripting, explains the powerful aspect of SSIS: scripting—developers can use scripting whenever other tasks or transformations can't help them to fulfill their requirements. There are two places for scripting in SSIS the—Script Task in Control Flow and Script Component in Data Flow. Scripting in both of these components will be covered in this chapter with samples.

Chapter 9, Deployment, describes how to deploy the developed packages and projects to a production environment, discussing different methods of deployment with the pros and cons of each way in real-world scenarios.

Chapter 10, Debugging, Troubleshooting, and Migrating Packages to 2012, explains the ability of SSIS to debug and troubleshoot like all robust systems. Developers need to know how to face problems in Control Flow or Data Flow, how to handle errors in Data Flow Task, and troubleshoot them. Debugging and troubleshooting have two sides in SSIS—Control Flow and Data Flow. This chapter describes both sides with appropriate examples. Also, this chapter has two recipes on migrating packages from the previous versions to 2012.

Chapter 11, Event Handling and Logging, explores all aspects of event handlers in SSIS besides logging in custom or built-in modes. SSIS provides a set of handlers for events on executable objects of Control Flow, which helps developers to handle these events and design appropriate operations on them. These event handlers also help developers to do some custom logging in their packages. There is a built-in logging feature in SSIS which can be used in general logging scenarios.

Chapter 12, Execution, covers different methods of package execution, and the properties and settings that can be configured at the time of execution.

Chapter 13, Restartability and Robustness, covers all these aspects of SSIS: SSIS has the structure to get input parameters from other applications. On the other hand, Packages can operate in a restartable mode. They can store their state at the time of failure and continue execution from that state next time. They are also capable of running Tasks in packages as a transaction.

Chapter 14, Programming SSIS, explains library classes for creating package and tasks, configuring them, deployment of a package, and running the package. Integration Services provide a set of .NET library classes and methods to do all parts of SSIS lifecycle operations from .NET programming.

Chapter 15, Performance Boost in SSIS, covers recommendations and best practices for raising the performance of packages and Data Flow. As an advanced part of each tool, there are some tips to raise the performance; they are described in this chapter.

1

Getting Started with SQL Server Integration Services

by Reza Rad and Pedro Perfeito

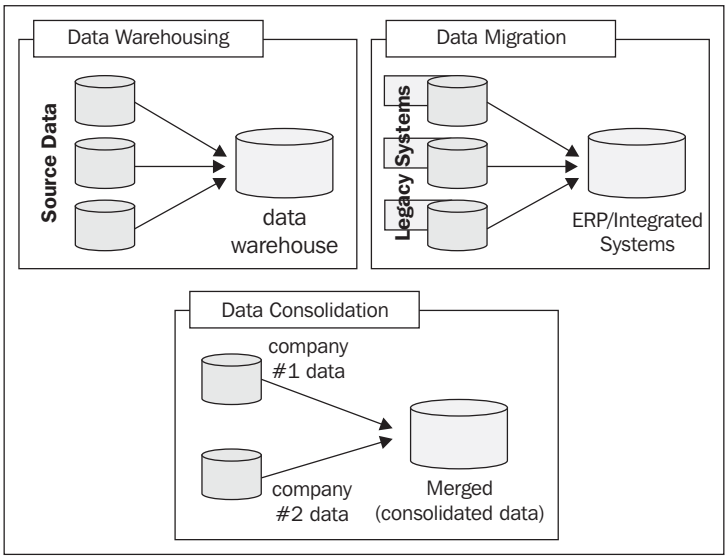
In this chapter, we will cover the following topics:

- Import and Export Wizard: First experience with SQL Server Integration Services (SSIS)
- Getting started with SSDT
- Creating the first SSIS package
- Getting familiar with Data Flow Task
- SSIS 2012 versus previous versions in Developer Experience

Introduction

As technology evolves, it is always necessary to integrate data between different systems. The integration component is increasingly gaining importance, especially the component responsible for data quality as well as the cleaning rules applied between source and destination databases. Different vendors have their own integration tools and components, and Microsoft with its SSIS tool is recognized as one of the leaders in this field.

SSIS can be used to perform a broad range of data integration tasks, and the most common scenarios are applied to Data Warehousing. The known term associated with Data Warehousing is the **Extract Transform and Load** (**ETL**) that is responsible for the extraction of data from several sources, their cleansing, customization, and loading into a central repository (for example, to a Data Warehouse, Data Mart, Hub, and so on). SSIS is also used in other scenarios, for example data migration and data consolidation. **Data Migration** is the one-time movement of data between databases and computer systems, and is needed when changes occur or when we upgrade our systems. **Data Consolidation** combines and integrates data from disparate systems and assumes high importance in a business environment with increasing acquisitions and mergers. The following diagram adapted from TDWI (`www.tdwi.org`) helps clarify the different scenarios where SSIS could be used:



New business challenges are driving organizations to adopt data integration projects. Some of these challenges are:

- Increasing demand for real-time information reporting and analysis
- Large volumes of data spread along the entire organization
- The need to comply with regulations, which often require to continuously track all changes to data and not just the net result of those changes

Although SSIS is an amazing tool for data integration, the same work can be done manually in almost all cases. As you can imagine, performing data integration tasks manually could be hard to maintain in terms of code, hard to scale properly, and would require more time to implement. From our perspective, since we have SSIS, there is no real reason to do it manually. The cost of ownership is not a problem either, because SSIS is included with SQL Server licenses that most organizations have already acquired.

In this chapter you will learn how to work with SSIS, how to create packages for data transfer, and you'll perform some simple operations with SSIS Package. At the end, we will highlight several improvements which are included in this new version.

As we will cover many recipes in this book, it is advisable to have **Adventure Works SQL 2012 sample database** installed.

Import and Export Wizard: First experience with SSIS

The Import and Export Wizard will be our first stop at SSIS. This wizard provides a simple ETL and is easy to use for basic data transfer operations. With this wizard you can choose a source, a destination, and map columns with few constraints on data transfer options. We will take a brief look at this wizard in our first experience with SSIS.

Getting ready

Install SQL Server 2012. SQL Server 2012 comes with three editions: Standard, Business Intelligence, and Enterprise. The Business Intelligence edition covers all requirements for this book that you'll need to install. With this edition you will have all SQL Server Integration Services features.

For many recipes in this book, you need to have the `AdventureWorks2012` and `AdventureWorksLT2012` sample databases installed. Information about installing these databases can be found in the book introduction.

To install sample databases, first download the database files from `http://msftdbprodsamples.codeplex.com` and then open SSMS to execute this statement (download `AdventureWorks2012` Data File and `AdventureWorksLT2012_Data`):

```
"" CREATE DATABASE AdventureWorks2012 ON (FILENAME = '<drive>:\<file
path>\AdventureWorks2012_Data.mdf') FOR ATTACH_REBUILD_LOG ;""
CREATE DATABASE AdventureWorksLT2012 ON (FILENAME = '<drive>:\<file
path>\AdventureWorksLT2012_Data.mdf') FOR ATTACH_REBUILD_LOG ;
```

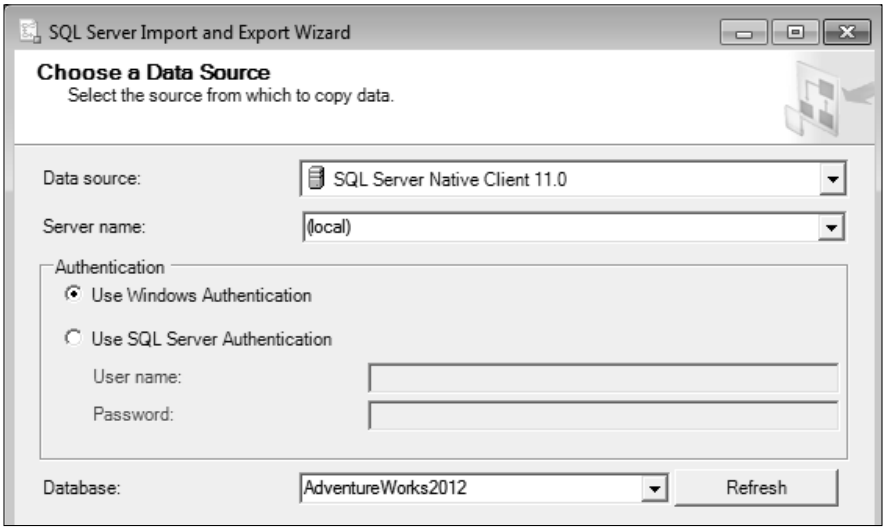
Note that you should replace the path of the file here with the path of the file downloaded on your machine.

Create a new database in SQL Server. Open **SQL Server Management Studio** (**SSMS**) from **Start Menu | Microsoft SQL Server 2012 | SQL Server Management Studio**. In the SSMS, connect to local computer instance and create a new database. Name this database as `PacktPub_SSISbook`.

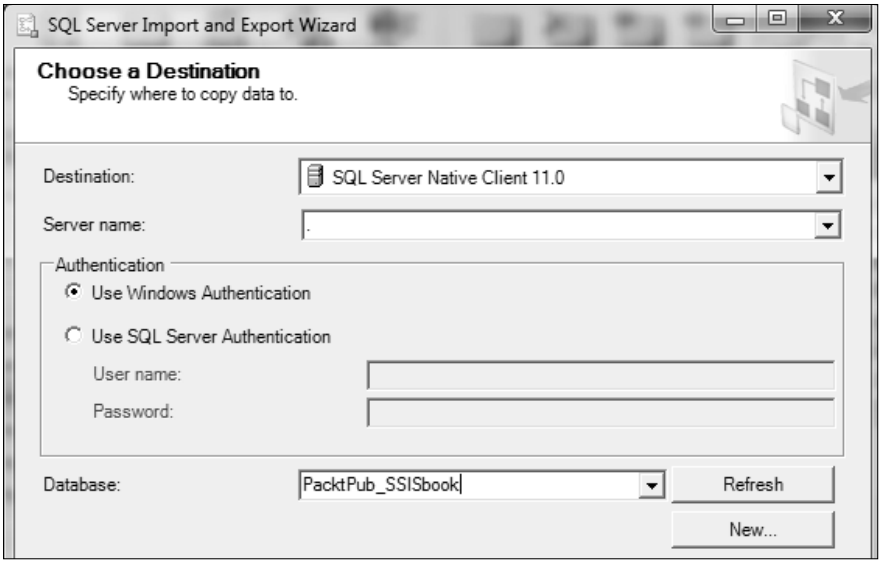
Note that you should run SSMS as administrator, to do this just right click on SQL Server Management Studio from path above and right click on it and choose Run as Administrator.

How to do it...

1. Open the Import and Export Wizard; there are three ways you could do it:
   - In the **Run** window, enter `DtsWizard`.
   - Open the wizard from the following address: **Start Menu | All Programs | Microsoft SQL Server 2012 | Import and Export Data**.
   - In SSMS, right-click on any database and then under **Tasks**, select *Import Data or Export Data*.
2. At the first step in the Import and Export Wizard, a welcome page will appear. Click on **Next** to enter the **Choose a Data Source** step, in this step you should choose where the **Data Source** comes from. The **Data Source** can be any source; from an Oracle database or SQL Server to any other database, flat files (such as `.txt` and `.csv`) or even Excel files, the range of source and destinations is based on data providers installed on the machine. For this sample, leave the **Data Source** option as its default option which is *SQL Server Native Client 11.0*.
3. We want to export two tables from the `AdventureWorks2012` database to another database. Therefore, leave the **Server name** as *(local)* or a single dot (.) or if you have a named instance you could use `.\<Instance-Name>`, and in the **Authentication** section leave the **Authentication Type** as *Windows Authentication*. This option will use your Windows account for connecting to the database, so obviously the Windows account should have read access to the underlying database.
4. In the Database drop-down box, select *AdventureWorks2012* from the list. Then click on **Next** and go to the next step.

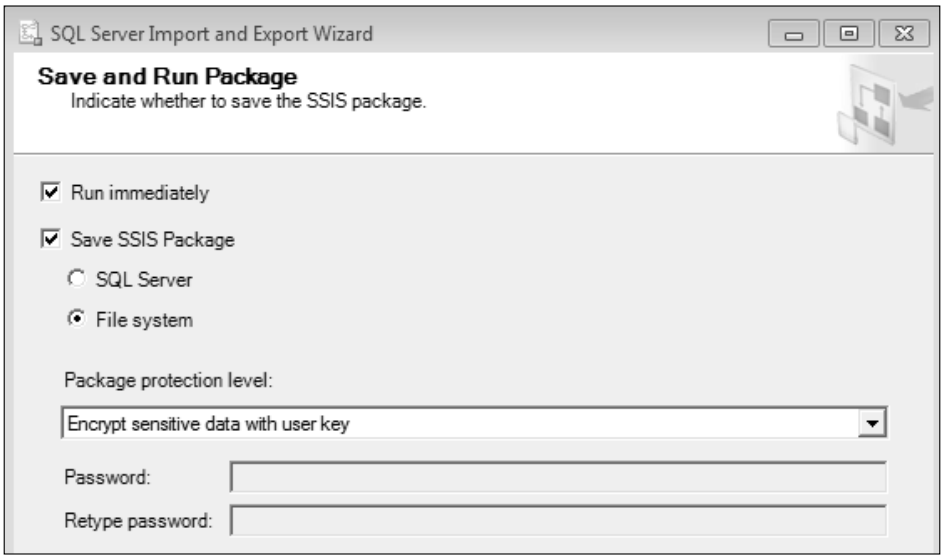


5. The next step is required to choose a **Destination**, therefore, provide the connection details of the data's destination (types of destinations can differ from databases to flat files). For this sample leave the destination as default value which is *SQL Server Native Client 11.0*.
6. Set the **Server Name** to *(local)* or dot (.) to connect default instance of current machine. Set the **Authentication** as *Windows Authentication*. Select *R1.1* in the **Database** drop-down list. Then click on **Next** to follow the wizard's steps.

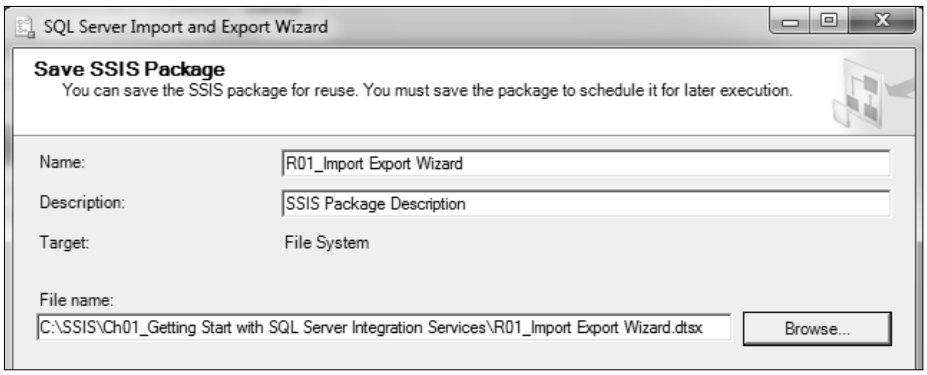


7. In the **Specify Table Copy or Query** step you can choose between selecting a table or view name for a source database or writing a query to fetch data from a source. For this example, choose the **Copy data from one or more tables or views** option.
8. Next, a list of tables and views from the source database will appear. For this example, select *HumanResource.Department*, *Person.Address*, and *Production.Product*. Then click on **Next**.

9. The **Save and Run Package** step is next; it provides the ability to save all the settings and configurations that you've set for an SSIS Package. We are going to save this package to see what the SSIS Package looks like. There are a lot of concepts and options associated with saving SSIS Packages which will be discussed in upcoming chapters, so don't worry about some terminologies here, all of them will be explored later. Check the **Save SSIS Package** option and click on the **Next** button.

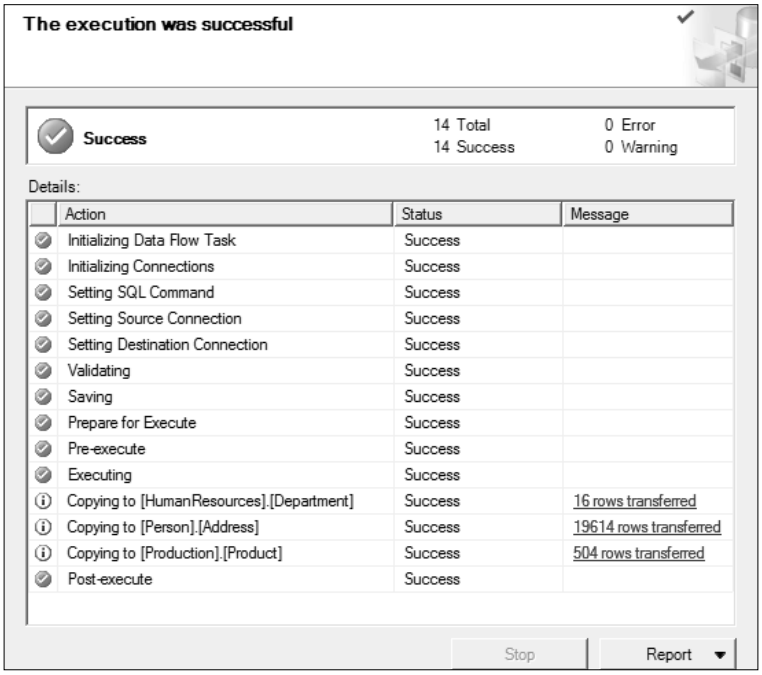


10. In the next step which is the **Save SSIS Package** dialog, type the name as `R01_ImportExportWizard`, and choose a location for the package file. Then click on **Next**.



11. Now a summary of all settings that you've done appears here; after reviewing the summary click on **Finish**.

12. After clicking on the **Finish** button, the Import and Export Wizard will show up and we can see all the messages generated during the package's execution. The number of rows copied are displayed, or any other information such as the number of rows transferred, validated, and any other actions.
13. Open and see the execution's report by clicking on the **Report** button.



14. Close the Wizard and open SSMS to check the destination database, you can see transferred tables there with data.

How it works...

In this recipe, we created the first SSIS Package with the Import and Export Wizard, this simple scenario exports some tables from the `AdventureWorks` database to an empty database. In the last few steps, we saved the whole data transfer scenario to an SSIS Package on a file system that we'll be able to open with **SQL Server Data Tools** (**SSDT**) in later recipes.

With the Import and Export Wizard you can import or export data from a source to a destination, this is the most simplistic ETL scenario. In the **Select a Data Source** step you perform the **Extract** part of ETL and fetch data from SQL server database (data source). The second step, which was the **Destination Select**, was configured during the **Load** part of ETL. **Load** indicates where data should be exported; we export data to a SQL Server database. In this first example scenario we don't have any specific **Transform** stage. We will see this part of the ETL later, during the *Data Flow* chapters of this book.

When you choose table(s) from the `AdventureWorks` database in the **Select Source Tables or Views** section, tables that don't already exist in the destination database will first be created.

After matching the columns and metadata, data will be transferred and a summary of all logs will show what happened during execution.

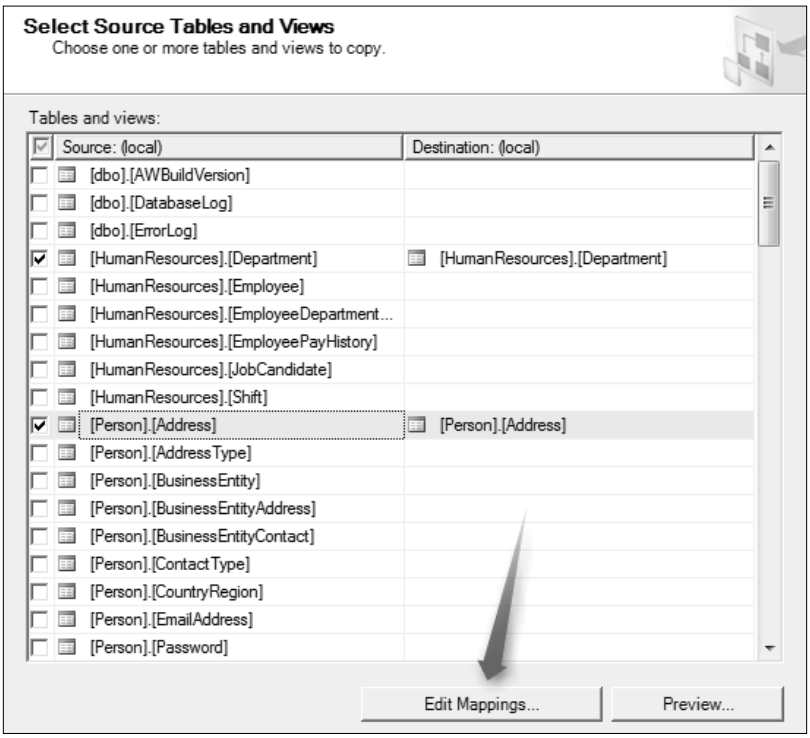
We saved this package to the file system; we can also save packages to an SQL Server. The difference between the different storage options for SSIS Packages with their pros and cons will be explored later in *Deployment* chapters.

There's more...

As you've seen so far, the Import and Export Wizard is a simple way to transfer data that covers our most basic requirements. But in real-world scenarios, you need some additional features, which we'll now discuss.

Mapping columns

In the **Select Source Tables and Views** step, when you select a table or view to transfer, an **Edit Mappings** button will be enabled. Note that you need to select a row in order to enable this button.

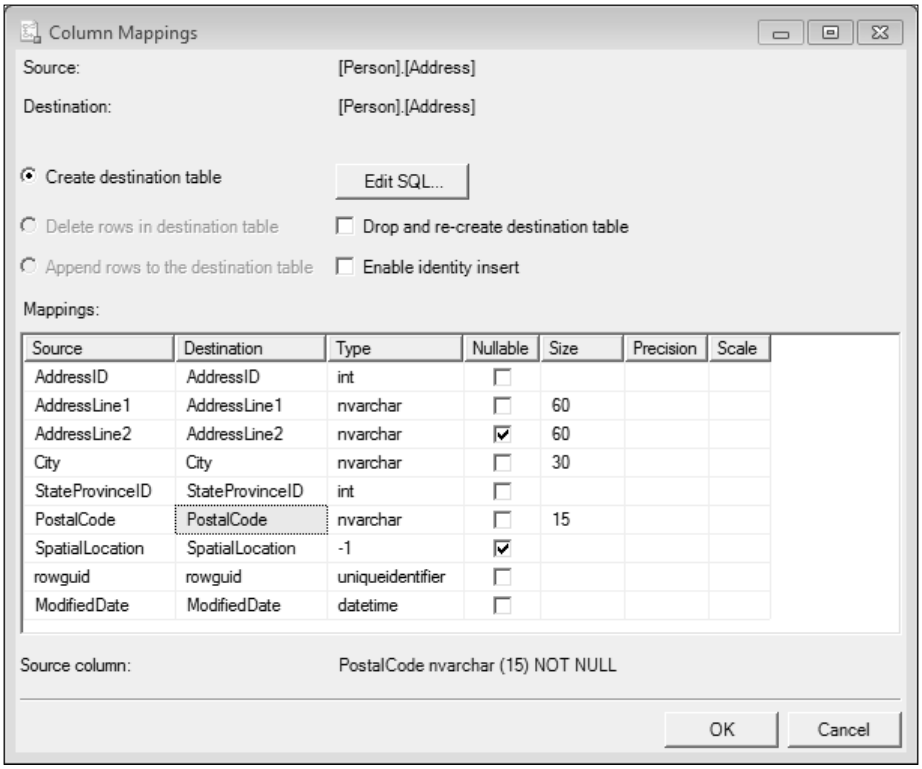


When you click on **Edit Mappings**, the **Column Mappings** window will appear. As you see, there are some options here for mapping **Source** and **Destination** columns.

When the destination table doesn't exist in the destination database, the **Create Destination Table** step will flag it. This means that the missing table will be created in the destination database; you can click on the **Edit SQL** button to see what the exact create table statement is; you can change the script as you want here.

When the destination table already exists in the destination database, the **Delete Rows** and **Append Rows** options in **Destination Table** will be selectable. You can select between deleting rows in a destination table before data transfer or appending new rows to existing records with these options.

The **Drop and re-create destination table** option will be selectable when the destination table already exists. Another important option is **Enable identity insert**, which should be checked if you load data into an `IDENTITY` column. The last part of the **Column Mappings** window is the **Mappings** section, which shows **Source** and **Destination** columns, as well as some additional column information. By default, all columns with the same name in **Source** and **Destination** will be mapped automatically. However, if the column names are different you should select columns by selecting the correct column name in the drop-down box. If you want to remove a column from data transfer you can simply choose the **<ignore>** option.



Configure transfer settings for multiple tables

In real-world scenarios, you need to configure transfer settings for all tables at once. Select multiple rows in the **Select Source Tables or Views Wizard** step by holding the *Ctrl* key and clicking on every row that you need, and then click on the **Edit Mappings** button. The **Transfer Settings** dialog box will open; all configurations that you set here will be applied to all selected tables.

In the **Destination schema name** you can choose a schema name from the destination database and use this schema for all selected tables. You can also type a schema name there; if that schema doesn't exist in the destination database it will be created and all tables will be created under this new schema. The **Drop and recreate new destination tables** option will be applied to all new tables, and the **Delete rows in existing destination tables** option will be applied to all existing tables. **Enable identity insert** is also applicable to all tables that have **Identity** columns.

Mapping data types

Data types which are automatically mapped through the **Column Mappings** window of the Import and Export Wizard are defined in XML files based on source and destination type. A list of all these XML files is available here:

```
<system drive>:\Program Files\Microsoft SQL Server\110\DTS\
MappingFiles
```

There is a mapping file for each source or destination, and details of data type mappings can be found there. The next screenshot shows a portion of the `MSSql8toOracle8` mapping file:

```
<?xml version="1.0" encoding="utf-8" ?>
- <dtm:DataTypeMappings
    xmlns:dtm="http://www.microsoft.com/SqlServer/Dts/DataTypeMapping.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    SourceType="SQLOLEDB;SQLNCLI*" MinSourceVersion="*" MaxSourceVersion="*"
    DestinationType="MSDAORA;OraOLEDB.Oracle;System.Data.OracleClient.OracleConnection"
    MinDestinationVersion="08.*" MaxDestinationVersion="*">
    <!--  smallint  -->
  - <dtm:DataTypeMapping>
    - <dtm:SourceDataType>
        <dtm:DataTypeName>smallint</dtm:DataTypeName>
      </dtm:SourceDataType>
    - <dtm:DestinationDataType>
      - <dtm:SimpleType>
          <dtm:DataTypeName>INTEGER</dtm:DataTypeName>
        </dtm:SimpleType>
      </dtm:DestinationDataType>
    </dtm:DataTypeMapping>
```

Querying the source database

If you need the ability to provide a custom query to read data from the source table(s), you can choose to write a query in order to specify the data to be transferred, and in the next step write the query to fulfill your requirements. You can also open a query from a file.

See also

- *Creating the first SSIS Package*
- *Getting familiar with Data Flow Task*

Getting started with SSDT

This recipe is an overview of **SQL Server Data Tools** (**SSDT**), where a user will spend most of his/her time while developing and maintaining SSIS projects.

This version is based on Visual Studio 2010, and the whole structure that supports the process of developing such projects has been significantly improved. Working with SSDT is not only easier for advanced users who require more flexibility, but also for beginners who can enjoy some new and interesting user interfaces to help them take their first steps with SSDT. Previous versions of SSIS used **Business Intelligence Development Studio (BIDS)** as their development environment.

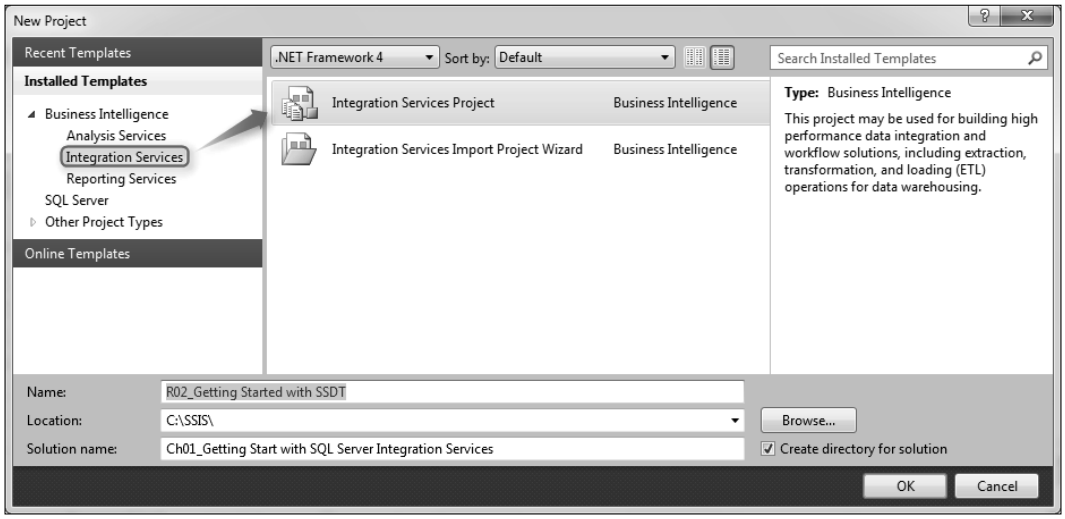
How to do it...

Open **SQL Server Data Tools** (**SSDT**) through the shortcut placed under Microsoft SQL Server 2012 or Open Microsoft Visual Studio 2010 under the Microsoft Visual Studio 2010 Start menu folders.

Once SSDT is open, a start page will be seen by default. The **Start Page** window contains useful information about the SSDT environment such as recently opened projects, links to create or open an existing project, and is also a useful area with several resources and the latest news to help stay up to date about several Microsoft platforms such as Windows, Web, Cloud, and so on.

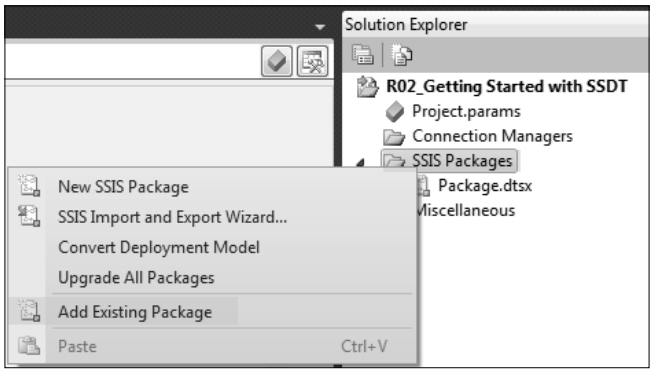
Now that SSDT is already opened, let's create a new SSIS project from the **Start Page** window in order to understand the basic steps as well as the remaining windows placed in the SSIS project example.

1. Click on **New Project...** and a Windows dialog will appear.

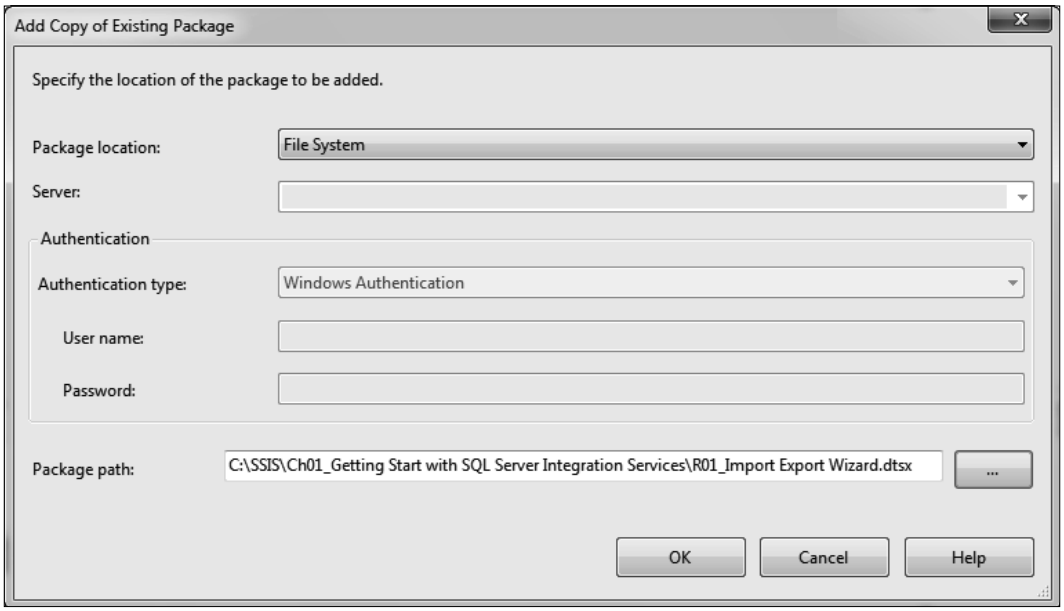


2. Under **Installed Templates**, expand **Business Intelligence** and click on **Integration Services**. In the center pane, select **Integration Services Project**.
3. Name the project as `R02_Getting Started with SSDT`. Name the solution as `Ch01_Getting Start with SQL Server Integration Services` in `C:\SSIS` and click on **OK**. An empty SSIS project will be created using the Project Deployment Model approach (default) with an empty package included.

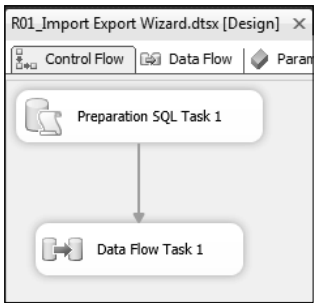
4. In the **Solution Explorer** pane, right-click on the **SSIS Package** folder, and choose *Add Existing Package*.



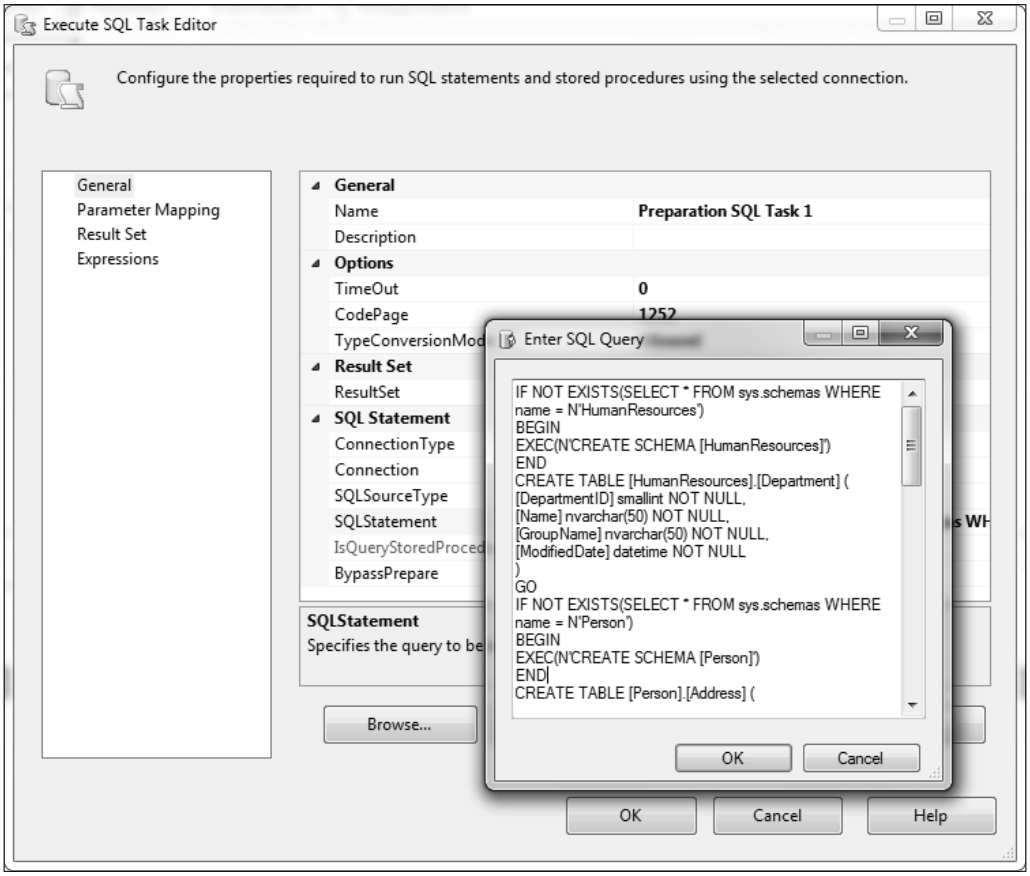
5. In the **Add Copy of Existing Package** dialog box, set **Package location** to *File System* and choose the package path from the file that you saved in the previous recipe from this address: `C:\SSIS\Ch01\Ch01R01_ImportExportWizard.dtsx`.



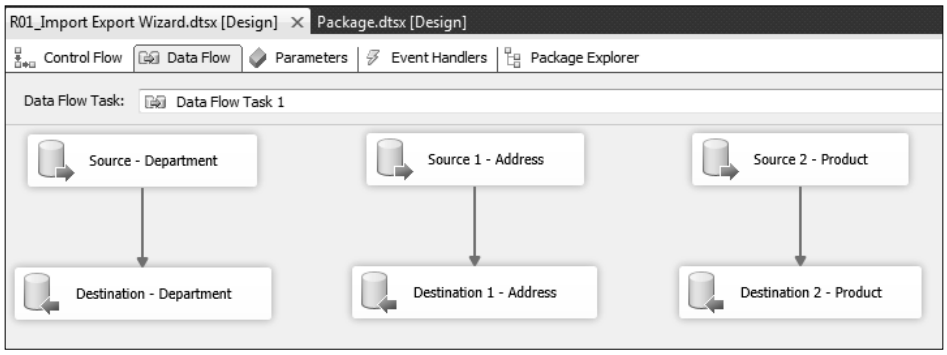
6. The new package will be added under the SSIS `Packages` folder, double-click on the package name in Solution Explorer to open it in Package Designer.



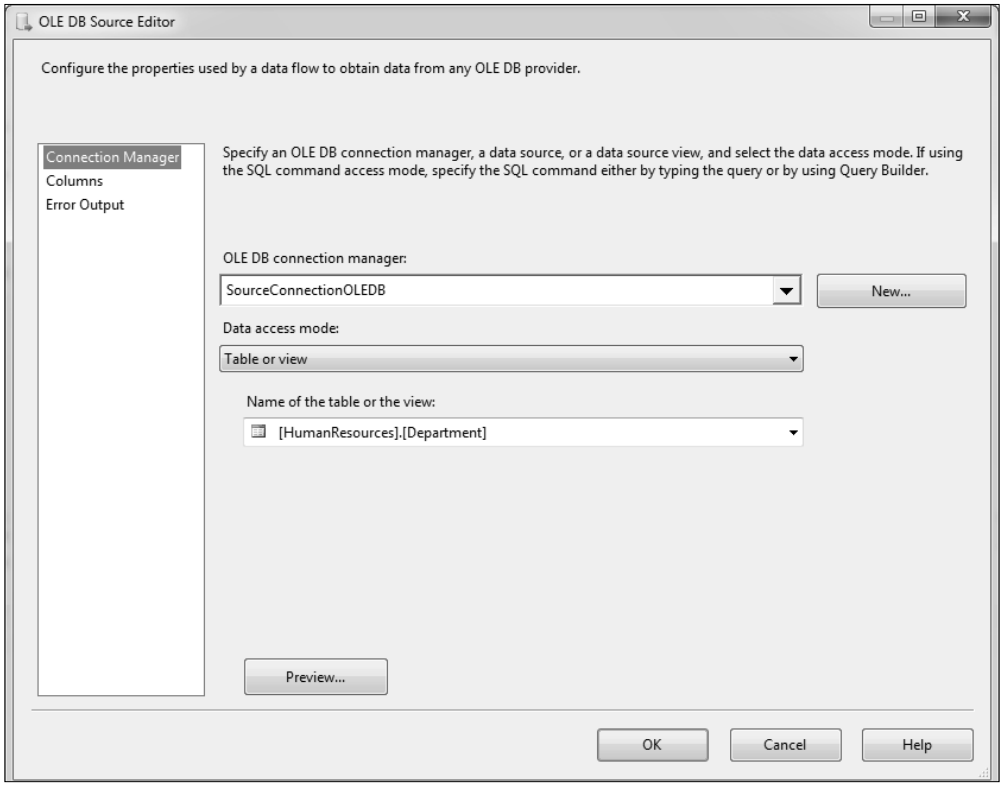
7. Double-click on **Preparation SQL Task 1** and the **Execute SQL Task Editor** dialog will open. Verify the **SQL Statement** property with a click on the ellipsis button in front of **SQL Statement**, and then close the editor.



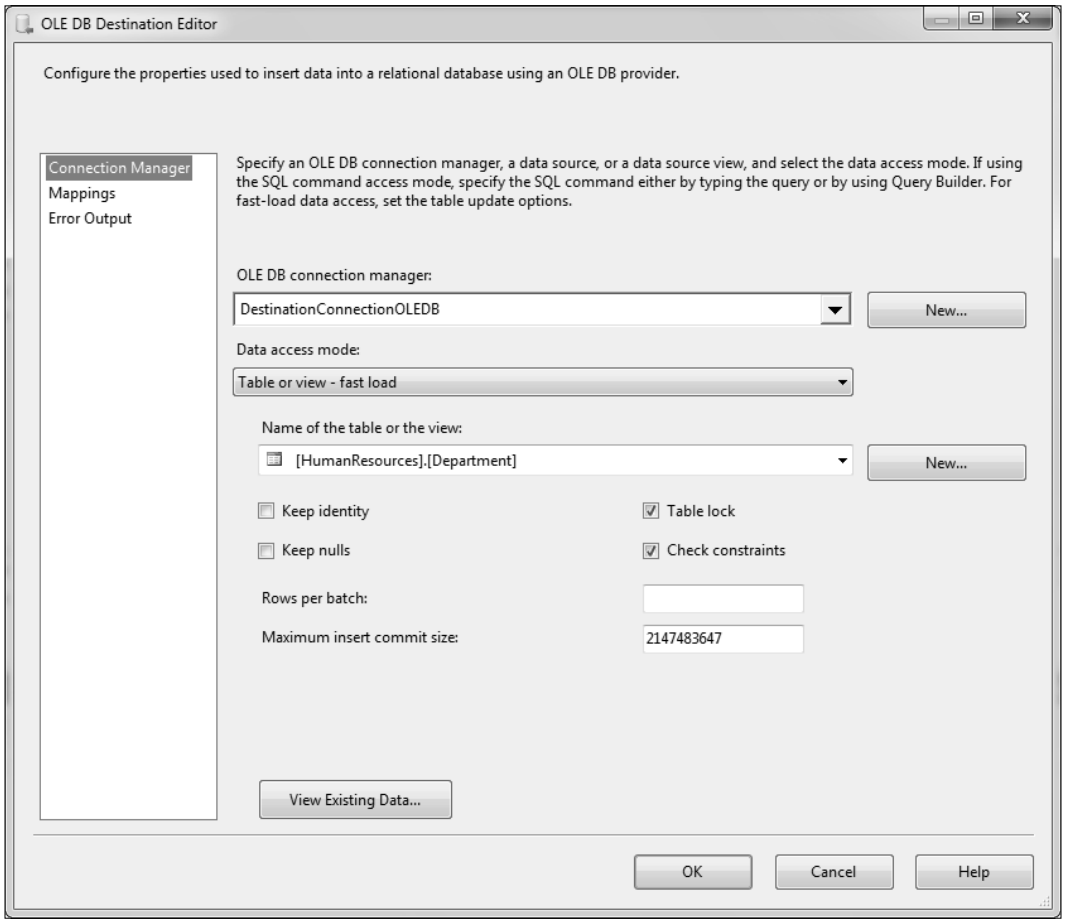
8. Double-click on **Data Flow Task 1**, and you will be redirected to the **Data Flow** tab, there are three source or destination combinations in **Data Flow**.



9. Double-click on the **Source-Department** component and the **OLE DB Source Editor** will open, verify the table name there.



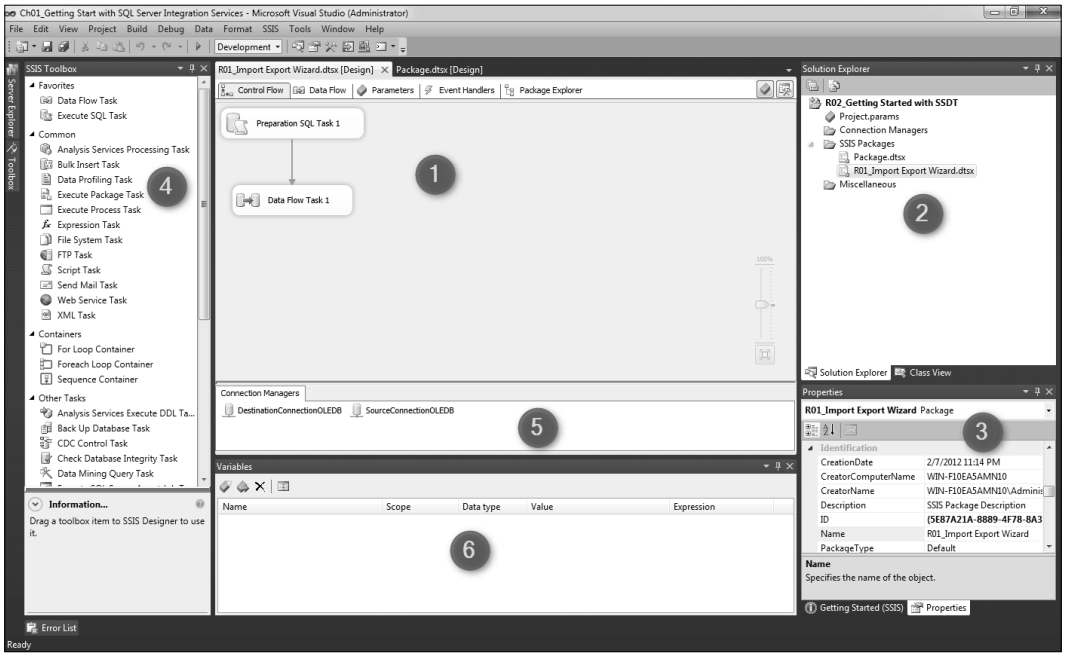
10. Double-click on **Destination-Department**, and in the **OLE DB Destination Editor**, verify the connection and table name.



The next recipe will explain the process of creating a new SSIS Package in more detail, and for that reason this recipe will focus on how we could get more value from SSDT to make the development and maintenance easier and faster.

How it works...

Now that the SSDT is open with an empty package, let's describe some of the windows that you should be familiar with, as shown in the next screenshot:



By default, SSDT creates a new and empty SSIS Package named `package.dtsx`. A **package** is a collection of SSIS objects including connection managers, tasks and components.

- **Package design area** (**1**)

  Control Flow is the most important tab; it's where a developer "explains" to SSIS what the package will do. The remaining tabs such as **Data Flow** (see recipe), **Parameters** (see *Chapter 11*, *Event Handling and Logging*), **Event Handlers** (see *Chapter 10*, *Debugging, Troubleshooting, and Migrating Packages to 2012*), the **Package Explorer** and **Progress** bar (available just at runtime) are also important and will be described in later recipes.

- **Solution Explorer** (**2**)

  The Solution Explorer section contains projects and their files.

  Each project consists of Project Parameters, Connection Managers, SSIS Packages, and the Miscellaneous folder.

  Project Parameters are parameters which are public for all packages in the project. We will discuss parameters in later chapters.

  The Connection Managers folder in the Solution Explorer consists of shared connection managers which are shared between all packages in a project.

  All SSIS Packages will be listed under the SSIS Packages folder.

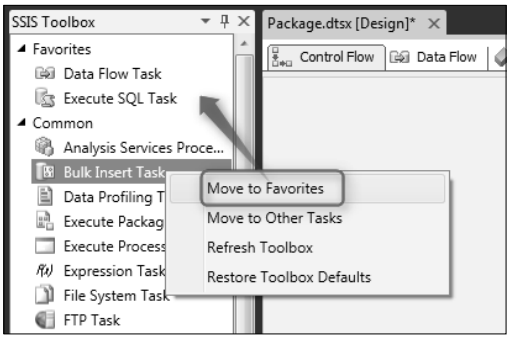
  The Miscellaneous folder can consist of any other files that are relevant to projects and packages, files such as documentation files, screenshots, and so on.

- **Properties panel** (**3**)

  In this panel, it's possible to read and edit the properties of each selected object in the Design area or Solution Explorer.

- **SSIS Toolbox** (**4**)

  In the SSIS Package, there are tasks and components which will be available depending on the tab selected in the Package design area. When the tab selected is **Control Flow**, the SSIS Toolbox will be grouped into four areas. The groups of tasks are the **Favorites**, **Common**, **Containers**, and **Other Tasks**. With these tasks, it's possible to control and inform SSIS about what should be done during execution. An interesting tip is that you can add tasks to the **Favorites** area anytime you like by right-clicking on each task and selecting **Move to Favorites**.



  Note that the SSIS Toolbox is completely different on the **Data Flow** tab; we will talk about it in later recipes.

- **Connection Managers** (**5**)

  Connection Managers are connections from the SSIS Package's components to source or destination data providers. There are different types of connection managers, some of them which are much in use are OLE DB Connection manager, Flat File Connection Manager and so on.

  Each connection manager can be used in one or more components in the SSIS Package to work with underlying data provider. Some data providers require the installation of special drivers to have connections to their data source.

  Each connection manager which is relevant to the current package will be listed in the Connection Manager's pane. Some connections are bold, these are referenced from a shared project's connection manager.

  We will discuss more about connections in the next recipes.

- **Variables Pane** (**6**)

  Each task in SSIS Package can send information to other tasks and it is possible by resorting to Variables. Package variables, their data types, their scope, and other properties exist in this pane, which will be described in greater detail in later chapters.

Creating the first SSIS Package

After understanding the SSDT environment, you will be able to make your first SSIS Package. Depending on the Data Integration project's complexity, it's always recommended to think carefully while selecting tasks and components to apply, as well as the order in which to execute them.

In this recipe, the first package created will read the number of records in an Adventure Works Microsoft Sample table and store it inside the SSIS Package.

Getting ready

You can reuse the recipe created in the previous section (adding a new package to it), or start from scratch as explained in the following steps:

1. Open **SQL Server Data Tools** (SSDT).
2. Click on **New Project...** and a Windows dialog will appear.
3. Click on the **Business Intelligence Projects** tab and under the **Installed Templates** section, select **Integration Services Project**.
4. Provide a name and location for the SSIS project and an empty structure as well as a package will be created.
5. In the **Solution Explorer**, select the empty *package.dtsx* and rename it to: `P01_FirstSSISPackage.dtsx`.

How to do it...

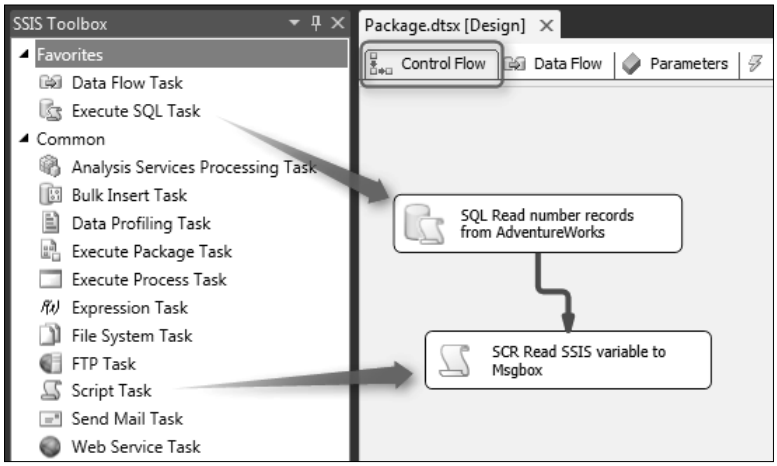
Now that the SSIS project is created, ensure that the empty package created by default is open and follow these steps:

1. Create an OLEDB Connection to the Adventure Works Microsoft sample database at the package level. (As already mentioned, if you create this connection in the Solution Explorer window, the connection will be created at the project level and will be automatically included inside all the existent packages).

2. In the **Configure OLE DB Connection Manager** Editor, select **New...** to create a new connection. If the connection already exists in the **Data Connections** list then select it here.

Ensure that the **Control Flow** tab is selected in the Package Designer area.

3. Drag-and-drop **Execute SQL Task** from **SSIS Toolbox** and place into the control flow design surface.
4. Double-click to edit **Execute SQL Task** or right-click and click on the **Edit** option.
5. Set the **Connection** property of the task to the connection created in step two.
6. Set the **Result set** to *Single row*.
7. Add the SQL Statement: `SELECT COUNT(*) AS NR_ROWS FROM SalesLT.Customer` to get the number of records from the `Customer` table.



8. Drag-and-drop **Script Task** from **SSIS Toolbox** and place into the Package Designer area and edit it by double-clicking on it.
9. Create a new variable to store the value provided by the previous task and add a message box inside the script (the script task is explained more clearly in *Chapter 11*, *Event Handling and Logging*) as follows:

```
MsgBox (Dts.Variables(0).Value, MsgBoxStyle.Information)
```

10. Run the package by pressing *F5*.

How it works...

This recipe created a basic SSIS Package that included the most used tasks among SSIS projects: the **Execute SQL Task**, to communicate with a database through SQL queries and the **Script Task** that allows us to create some custom "work" when other tasks fall short. Naturally, it is too early to go into details with these two tasks, but this way it's possible to have an initial look at these common tasks.

Getting familiar with Data Flow Task

While the **Control Flow** tab under the Package Designer is where the main workflow of the package is manipulated, the **Data Flow** tab introduced in this recipe is the place where data is transformed and moved between a source and destination. The **Data Flow** tab under the Package Designer is used to create or edit such transformations and movements for each **Data Flow** task which is placed in the **Control Flow** tab.

Getting ready

It's possible to have several Data Flows in the Control Flow, but the order of execution for those Data Flows should be planned carefully. To be familiar with the Data Flow task, this recipe introduces a simple but very common scenario of incorporating the content of an Excel file into a database, a SQL Server database for example.

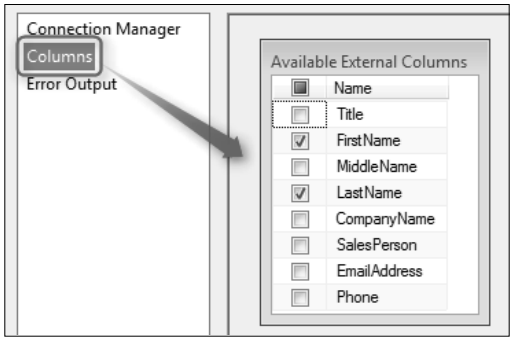
1. Open **SQL Server Data Tools** (SSDT).
2. Create a new project and provide a name and location for it.
3. Open the empty `package.dtsx` created by default and rename it to `P01_DataFlowTask.dtsx`.

How to do it...

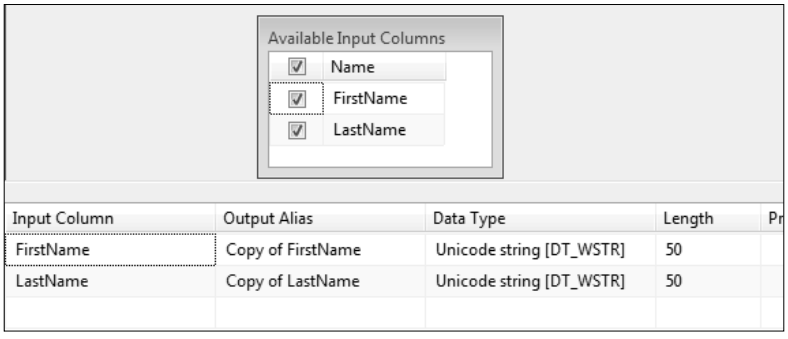
Imagine a case where SSIS needs to periodically integrate data produced by an external system, and this data needs to be prepared to fit the destination requirements such as schema and data values exactly. To accomplish this task, we need to read data from an Excel worksheet, perform data conversions, verify whether data already exists at the destination, and finally insert into an SQL table at the destination.

1. In the Package Designer select the **Control Flow** tab.
2. Drag-and-drop a **Data Flow** task from the **SSIS Toolbox** to **Control Flow**.
3. Open **Data Flow** for editing by double-clicking under the task or just right-click and select **Edit**.

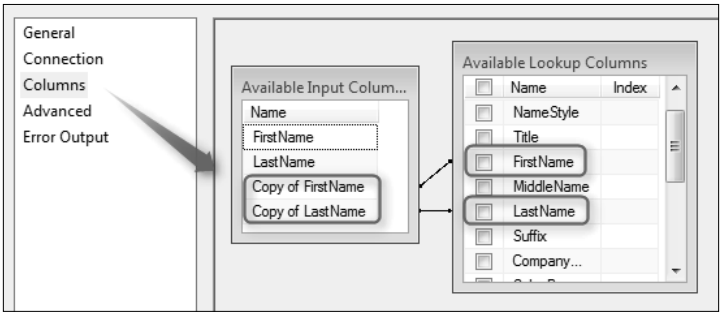
4. Make sure that the **Data Flow** tab is selected in the Package Designer. At this step **Data Flow** is naturally empty.
5. Because we need to read some data from an Excel file, simply drag-and-drop the **Excel Source** component (under the **Data Sources** group) from **SSIS Toolbox** and place it into the Package Designer area. Note that **SSIS Toolbox** has different content listed; it has components in spite of tasks that exist when the **Control Flow** is selected in the Package Designer.
6. Double-click on the **Excel Source** component and click on the **New** button to create a connection to the source Excel file.
7. Set the Excel file's path to `C:\SSIS\Ch01_Getting Start with SSIS\FILES\R04_NewCustomers.xlsx` and click on **OK**.
8. In the **Excel Source Editor**, set the name of the Excel sheet to `Sheet1$`.
9. Select the **Columns** tab and choose the columns **Firstname** and **Lastname**, which will be used along the data flow.



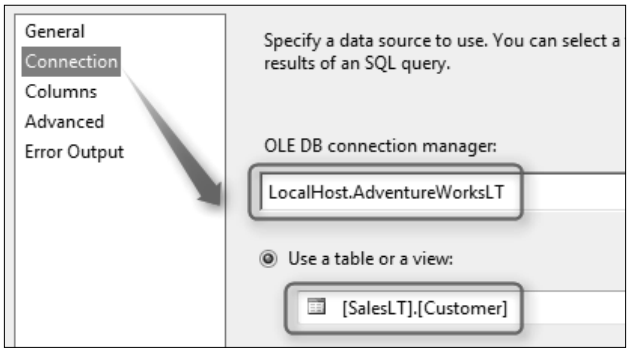
10. Click on **OK** to finish editing **Excel Source**.
11. Drag-and-drop **Data Conversion** from **SSIS Toolbox** to the Package Designer.
12. Select the column's **FirstName** and **LastName**, and change the **Length** for each to 50 characters.



13. Drag-and-drop the **Lookup** component from **SSIS Toolbox** into the Package Designer.
14. Maintain all the properties with default values and create a connection to an SQL destination database (`AdventureWorksLT`).
15. In the list to select table or view, select the destination table *'SalesLT'.'Customer'*.
16. Map the source converted columns **Copy of FirstName** and **Copy of LastName** to the destination SQL columns.

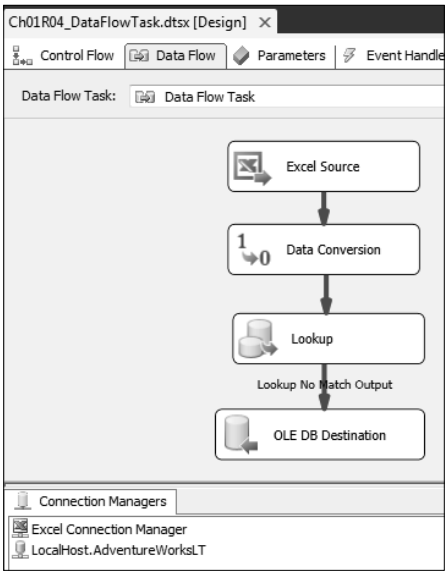


17. Click on **OK** to finish editing the lookup.
18. Drag-and-drop the **OLE DB Destination** component from **SSIS Toolbox** into the Package Designer in order to load data into SQL.
19. Link the output **No Match Rows** from **Lookup** to insert into destination only those records that don't yet exist at the destination.
20. Edit the destination component and set the SQL connection and also the destination table; maintain all the default values for the remaining controls.



21. Click on **OK** to finish editing the **OLE DB Destination** component.

22. Run the package by pressing *F5*.



How it works...

This Data Flow reads some customer data (first name and last name) from an Excel file, applies some common transformations and inserts the data into an SQL table named `SalesLT.Customer`. Some transformations are usually applied to make source data fit the destination's requirements. In this example, data is converted and we verified whether the current data in the incoming rows already exists at the destination (the purpose is to avoid data duplication).

Detailed descriptions about each source (**Transformation** and **Destination**) used in this example will be explored in later chapters. In this recipe, we just need an overview of **Data Flow**.

SSIS 2012 versus previous versions in Developer Experience

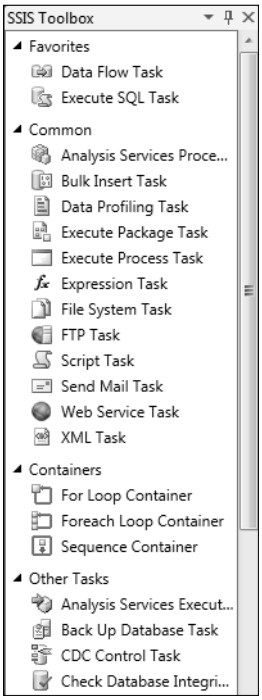
This book aims at the new version of SSIS, which is SSIS 2012. There are a bunch of changes in SSIS 2012, this recipe covers some of the differences between SSIS 2012 and previous versions. The SSIS 2012 changes aren't limited to SSDT and design changes alone, there are also many changes while interacting with outside packages and package deployment, as well as new tasks and transformations; all of which we will explore in different recipes of this book in appropriate case scenarios. For this recipe, differences in SSDT as compared to previous versions are highlighted.

Getting ready

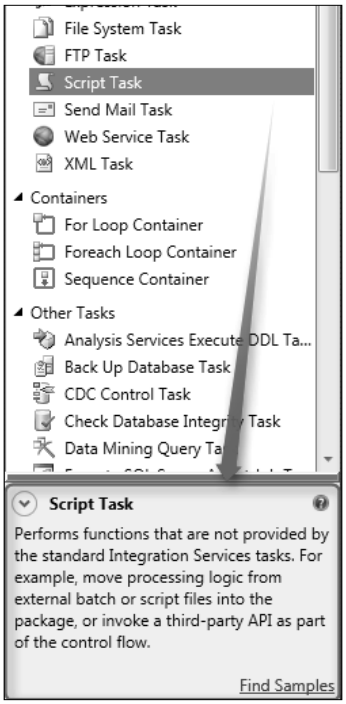
As this recipe compares the two versions, having SSIS 2012 and SSIS 2008 installed can be useful. SSIS 2012 is required but 2008 is optional (will help in comparison).

How to do it...

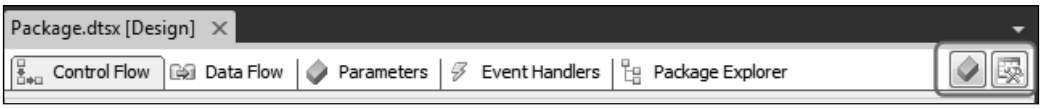
1. Create a **New SSIS Project**.
2. The first difference you'll notice in SSIS 2012 is that the SSDT is Visual Studio 2010 with a lot of improvements in the Editor. Earlier versions of SSIS work with previous versions of Visual Studio. SSIS 2008 worked with Visual Studio 2008, and SSIS 2005 worked with Visual Studio 2005.
3. When a new Package is created, the SSIS Toolbox is shown in the SSDT. As you would notice in the SSIS Toolbox, there are some categories that differ from the previous SSIS 2008 version. The previous version had only three sections: **Control Flow Items**, **Maintenance Plan Tasks**, and **General**. In SSDT 2010, sections are organized as: **Favorites**, **Common**, **Containers**, and **Other Tasks**. **Favorites** contain tasks that can be moved in here by right-clicking on items and then clicking on **Move to Favorites**. **Common** consists of a list of tasks that is among the most useful tasks. **Containers** have their own section in this version of SSDT, and all other tasks are placed in the **Other Tasks** section.



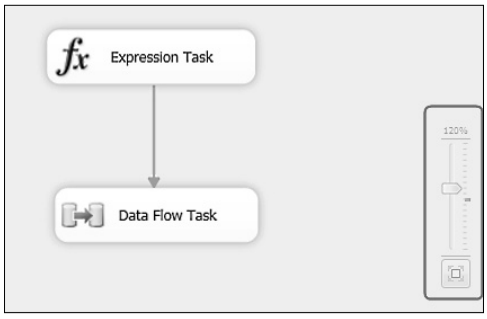
4. A quick description of each task or container is available under the **SSIS Toolbox**, as you can see in the following screenshot:



5. There are two icons for switching between the **Toolbox** pane and the **Variables** pane in Package Designer.



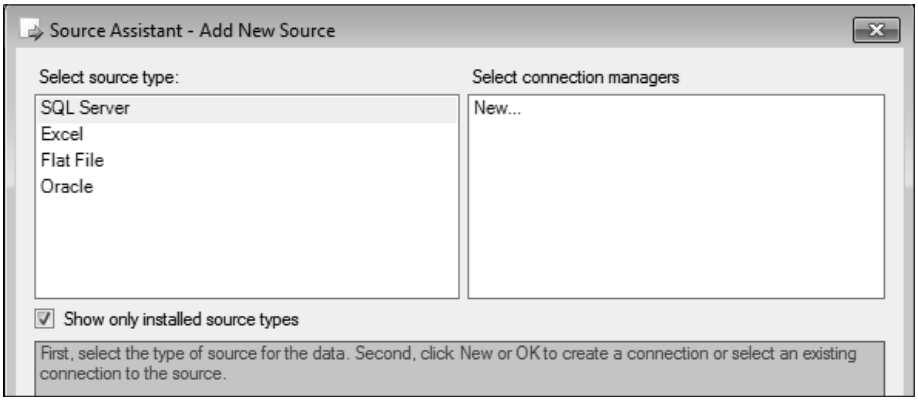
6. There is a scroll bar magnifier on the **Control Flow** tab which adds the ability to magnify a view of Control Flow, you can also choose auto fit.



7. There is a new tab in the SSDT 2012 Package Designer named **Parameters**; we will discuss this Parameters tab in later chapters.
8. There is also an **Undo** icon; SSIS 2008 and 2005 suffered from the lack of undo operations, but in SSDT 2012 you can undo operations with *Ctrl + Z* or by clicking on the undo icon.
9. In the **Solution Explorer**, create a new **Project Level** connection. This is a shared connection which can be seen in all packages' connection manager's pane within the project.
10. Right-click on the blank **Control Flow** area and select **Getting Started**; you will see the new **Getting Started** pane on the right-hand side at the bottom of SSDT, which shows some help and links as seen in the following screenshot:

11. Drag-and-drop a **Script Task** from toolbox to control flow, and double-click on Script Task. In the **Script Task Editor**, in the **Script Language** property, you can select between *Visual C# 2010* and *Visual Basic 2010* both of which are under the .NET 4.0 Framework. We will explore all details about scripting SSIS in later chapters.
12. Go to the **Data Flow** tab, you will see this sentence: **No Data flow task has been added to this package. Click here to add a new data flow task**. Just click on the link and a new empty data flow will be created.

13. Notice that there are two new options **Source** and **Destination** in the **SSIS Toolbox** which can be found under the **Favorites** sections **Source Assistant** and **Destination Assistant**. With these assistants you can simply select a data source or destination as you want with a single assistant component.



How it works...

We had already reviewed some changes to SSDT Design in the **Control Flow** and **Data Flow** tab areas. There are lots of other changes such as better annotations, grouping components in data flow, and many other improvements in the UI which we will explore with the many examples in this book in later recipes.

On the other hand, there are some major changes like new tasks and transformations, like Expression Task, CDC Control Task, Data Flow Components, and DQS Cleansing transformation which we will explore in appropriate chapters.

There are major changes in the deployment of packages and working with packages from outside and package execution, which will be explored in package deployment chapters later on.

Where to buy this book

You can buy Microsoft SQL Server 2012 Integration Services: An Expert Cookbook from the Packt Publishing website: `http://www.packtpub.com/microsoft-sql-server-2012-integration-services-expert-cookbook/book`.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information:
www.packtpub.com/microsoft-sql-server-2012-integration-services-expert-cookbook/book